Réseaux de neurones en traitement d'images

Gilles Burel

Thèse de doctorat de l'Université de Bretagne Occidentale

Spécialité: Electronique

Présentée par: Monsieur **Gilles BUREL**

RESEAUX de NEURONES en TRAITEMENT d'IMAGES: Des Modèles Théoriques aux Applications Industrielles

Soutenue le 6 décembre 1991, devant la commission d'examen:

Président:	*G. Stamon*	*Professeur à Paris V - Sorbonne*
Rapporteurs:	*L.C. Calvez*	*Professeur à l'Université de Brest*
	C. Jutten	*Professeur à l'Université de Grenoble*
Examinateurs:	*P. Jarry*	*Professeur à l'Université de Bordeaux*
		Directeur de thèse
	J.Y. Catros	*Responsable du laboratoire RII de Thomson CSF-LER*
	B. Dorizzi	*Professeur à l'INT (Evry)*
	J.P. Cocquerez	*Professeur à l'ENSEA (Cergy)*

.

à mes parents

RESUME

Les travaux présentés portent sur les réseaux de neurones appliqués au traitement du signal et de l'image. On se place d'emblée du point de vue de l'industriel impliqué dans la recherche, c'est à dire que l'on s'intéresse à des problèmes réalistes, sans pour autant négliger la recherche théorique.

Dans une première partie, nous montrons l'intérêt des réseaux de neurones comme source d'inspiration pour la conception de nouveaux algorithmes. Nous proposons en particulier une structure originale pour la prédiction, ainsi que de nouveaux algorithmes de Quantification Vectorielle. Les propriétés des algorithmes existants sont également éclaircies du point de vue théorique, et des méthodes de réglage automatique de leurs paramètres sont proposées.

On montre ensuite les capacités des réseaux de neurones à traiter un vaste champ d'applications d'intérêt industriel. Pour divers problèmes de traitement de l'image et du signal (de la segmentation à la séparation de sources, en passant par la reconnaissance de formes et la compression de données), on montre qu'il est possible de développer à moindre coût une solution neuronale efficace.

REMERCIEMENTS

Je tiens en tout premier lieu à remercier Monsieur Boyer, Directeur de Thomson CSF-LER, et Monsieur de Couasnon, Directeur technique, pour m'avoir accueilli dans leur établissement, et m'avoir permis de réaliser ce travail.

Monsieur Stamon, Professeur à Paris V - Sorbonne, me fait l'honneur d'assurer la présidence du jury d'examen. Je voudrais lui exprimer ici ma plus sincère reconnaissance.

Monsieur Jarry, actuellement Professeur à l'Université de Bordeaux, et responsable du Laboratoire d'Electronique et Systèmes de Télécommunications de l'Université de Brest (L.E.S.T.) jusqu'en juillet 1991, a été mon Directeur de thèse, et a assuré le suivi de ce travail. Je lui adresse mes plus sincères remerciements, ainsi que pour la confiance qu'il m'a témoignée en me confiant le cours de traitement d'images du D.E.A. d'Electronique dont il a la responsabilité.

Mon intérêt pour le traitement du signal date de l'époque où j'ai eu la chance de suivre l'excellent cours de Monsieur Calvez, Professeur à l'Université de Brest. Je voudrais le remercier vivement d'avoir accepté d'examiner ma thèse en tant que rapporteur.

Les travaux de Monsieur Jutten, Professeur à l'Université de Grenoble, ont toujours été à l'avant-garde de la recherche en réseaux de neurones. Je lui suis particulièrement reconnaissant de l'honneur qu'il me fait en acceptant de participer au jury en tant que rapporteur.

Monsieur Catros, responsable du laboratoire de Reconnaissance et Interprétation en Imagerie (RII) de Thomson CSF-LER, a constamment suivi l'évolution de ce travail, et m'a fait profiter de sa riche expérience du traitement d'images. Je voudrais ici lui exprimer ma plus vive gratitude.

J'adresse mes plus vifs remerciement à Madame Dorizzi, Professeur à l'INT, et responsable du département Intelligence Artificielle, pour sa participation au jury. Dans le cadre du projet Esprit II "Pygmalion", sa rigueur scientifique et ses qualités de management ont guidé mes premiers travaux en réseaux de neurones.

Monsieur Cocquerez, Professeur à l'ENSEA, anime une dynamique équipe de Traitement d'Images. Je voudrais lui exprimer ici mes plus sincères remerciements pour sa participation au jury.

Je ne saurais oublier mes collègues des L.E.R., qui ont directement ou indirectement contribué à ce travail. Je voudrais remercier tout particulièrement Isabelle Pottier, pour sa contribution essentielle à ce travail par ses critiques constructives et son aide précieuse dans la réalisation des logiciels d'expérimentation, ainsi que Dominique Carel, pour sa contribution à l'application de reconnaissance de pièces industrielles 2D (notamment la comparaison avec une méthode symbolique), et Jean-Ronan Vigouroux pour diverses discussions enrichissantes.

J'ai eu plaisir à encadrer les stages de Hugues Henocq (synthèse de textures), Isabelle Pottier (logiciel d'apprentissage supervisé, et compression d'images), et Jean-Paul Accarie (chiffres manuscrits). Qu'ils soient remerciés pour l'aide qu'ils m'ont apportée.

INTRODUCTION

On assiste depuis 1986, avec la parution de l'ouvrage collectif "Parallel Distributed Processing"[*Rumelhart*86], à un regain d'intérêt pour les modèles de traitement de l'information inspirés du fonctionnement du système nerveux. Cette voie de recherche n'est pourtant pas nouvelle, car elle a son origine dans des travaux conduits à la fin des années 40 (modèle de Hebb pour l'apprentissage [*Hebb*49]). Dès les années 60, un modèle connu sous le nom de "perceptron" [*Rosenblatt*62] a été appliqué à la reconnaissance de caractères. Mais les limitations de ces premiers modèles ont provoqué un rapide désintérêt.

Le succès de l'ouvrage "Parallel Distributed Processing" est dû à la présentation de nouveaux algorithmes, plus performants, et à la mise en valeur des avantages et particularités des systèmes nerveux biologiques (parallélisme massif, capacités d'apprentissage, mémoire distribuée, résistance aux dégradations, ...). Des applications à fort potentiel médiatique ont rapidement vu le jour, comme par exemple le système "NET-TALK" [*Sejnowski*87], qui apprend à lire à haute voix du texte en anglais. Les modèles de traitement de l'information inspirés du fonctionnement du système nerveux sont regroupés sous le nom générique de "réseaux de neurones" (ou "modèles connexionnistes").

De nombreuses conférences relatives à ce domaine ont déjà eu lieu, et un projet Européen a débuté en 1989 (Esprit Project n^o 2059 "Pygmalion"). Le nombre de sociétés industrielles et de laboratoires universitaires s'intéressant aux techniques neuronales est également en constante augmentation. On assiste actuellement dans ce domaine à un foisonnement de modèles et d'applications.

La présente thèse a été réalisée aux Laboratoires Electroniques de Rennes de Thomson CSF, l'un des deux laboratoires centraux de Thomson, dont une des missions est de réaliser des recherches de pointe dans les domaines du traitement du signal et de l'image. Les remarquables performances du système visuel humain laissent entrevoir l'apport que l'on peut espérer de modèles d'inspiration biologique. Pourtant, la plupart des modèles proposés ne sont pas exploitables au vu des performances technologiques actuelles (et ne le seront probablement pas avant des dizaines d'années). Par exemple, certains modèles proposés dans [*Grossberg*88] sont définis par des équations différentielles. Il sont donc intéressants pour comprendre certains aspects du fonctionnement cérébral, mais sont difficilement applicables à des problèmes réels du fait de la lenteur d'une telle simulation.

Notre objectif est ici de montrer la capacité de certains modèles neuronaux à

traiter des problèmes réalistes de reconnaissance de formes ou d'analyse du signal, ainsi que leur intérêt comme source d'inspiration pour mettre au point de nouveaux algorithmes. Plus précisément, nous nous attacherons à répondre aux points suivants:

- Améliorer ces modèles pour faciliter le traitement d'une vaste panoplie d'applications.
- Contribuer à conforter leurs bases théoriques.
- Résoudre par une approche neuronale des problèmes de traitement d'image d'intérêt industriel.
- Proposer de nouveaux algorithmes. Par exemple, on proposera au chapitre 10 un nouvel algorithme de séparation de mélanges de signaux qui est capable de séparer des mélanges non-linéaires et ne nécessite aucune connaissance a priori sur les sources.

L'intérêt industriel ne peut être atteint qu'en gardant à l'esprit les limitations technologiques présentes. Ainsi, le nombre de neurones simulables sur un ordinateur de puissance moyenne se chiffre en milliers, alors que le cerveau humain contient de l'ordre de 10^{11} neurones. C'est pourquoi, nous ne suivrons pas la voie préconisée par de nombreux auteurs, et qui consiste à traiter directement une image par un réseau de neurones, comme c'est le cas dans le système nerveux. En effet, la "faiblesse numérique" des neurones simulés ne nous semble pouvoir être compensée qu'en faisant appel aux connaissances du traitement d'image pour réaliser des prétraitements qui permettent de réduire le volume de données à traiter et d'introduire certaines invariances. Nous verrons que pour les problèmes de classification, la puissance des techniques neuronales permet de mettre en œuvre des prétraitements réversibles au sens du problème à traiter, ce qui élimine toute perte d'information pertinente dans le prétraitement (et tout risque de recouvrement de classes dans l'espace paramétrique lorsque l'on traite un grand nombre de données).

On pourrait objecter que la recherche concernant la réalisation matérielle de machines neuronales est actuellement très active. Mais il convient de ramener les espoirs à de justes proportions, car ces machines permettraient tout au plus de gagner quelques ordres de grandeur. De plus, la recherche dans ce domaine semble plutôt s'orienter [*Saucier*89] vers des machines neuronales qui ne réalisent pas l'apprentissage (celui-ci devant être réalisé à part sur un ordinateur classique).

Le travail relaté dans cette thèse a donné lieu à plusieurs brevets et publications. Les lecteurs qui ne sont pas familiers avec les techniques neuronales pourront lire en introduction notre article "Réseaux de neurones en traitement d'images"[*Burel*90], qui donne un bref aperçu de ce domaine et d'une partie des

études décrites ci-après.

Les chapitres 1 et 2 situent le contexte. Dans le chapitre 1 nous présentons certains aspects des connaissances actuelles concernant le fonctionnement du système nerveux. Les modèles neuronaux que nous utilisons peuvent être décrits en termes purement mathématiques, et ne sont pas directement calqués sur la réalité biologique. Il eût toutefois été injuste de passer sous silence ces connaissances biologiques, qui ont été une source d'inspiration importante au début de nos travaux. Le chapitre 2 décrit les modèles neuronaux que nous utiliserons et leurs algorithmes d'apprentissage associés. Il tente également de situer ces modèles par rapport à des théories plus classiques.

Le chapitre 3 présente les améliorations que nous avons apportées aux modèles et algorithmes utilisés pour faciliter le traitement de problèmes réels. Il s'agit principalement de méthodes de réglage automatique des paramètres de l'apprentissage, et de méthodes de controle de la saturation.

Dans le chapitre 4 nous proposons de nouveaux algorithmes et résultats théoriques. La première partie du chapitre présente une nouvelle approche pour les réseaux de neurones multi-couches: la Représentation Scalaire Distribuée. Cette approche permet d'accroître de façon importante la capacité du réseau à traiter des problèmes non-linéaires. Puis, dans une seconde partie, nous proposons de nouveaux algorithmes de Quantification Vectorielle qui présentent d'intéressantes propriétés de conservation de la topologie.

Les chapitres 5 à 9 présentent diverses applications de traitement d'images. Ces applications ont été choisies pour couvrir un vaste champ de problèmes : la segmentation (chap. 5), la reconnaissance de formes (chap. 6 et 7), la reconnaissance de l'écriture (chap. 8), et la compression d'images (chap. 9):

- Le chapitre 5 présente une nouvelle approche pour la discrimination de texture, qui met en œuvre un prétraitement réversible au sens de la texture suivi d'un classifieur neuronal. Cette approche est validée dans les domaines visible et Infra-Rouge.

- Les chapitres 6 et 7 présentent des applications de reconnaissance de forme (en 2D avec occlusions, et en 3D) dans le domaine de la robotique. En 3D, on utilise un prétraitement réversible au sens de la forme.

- Dans le chapitre 8, on présente une application à la reconnaissance de chiffres manuscrits. Une méthode neuronale très performante a déjà été proposée antérieurement [*LeCun*89]. Nous montrerons comment il est possible de réduire considérablement la complexité du système de reconnaissance en mettant en œu-

vre un prétraitement quasi-réversible au sens du chiffre. Comme ce prétraitement est quasi-réversible, il n'introduit pas de perte d'information significative.

- L'objet du chapitre 9 est la compression d'image. Nous montrerons comment les propriétés topologiques d'un algorithme neuronal particulier peuvent être exploitées pour améliorer les taux de compression.

Enfin, dans le chapitre 10, nous proposons un nouvel algorithme pour un problème fondamental en traitement du signal : la séparation de mélanges de signaux. Cet algorithme est capable de traiter des mélanges non-linéaires, et ne nécessite aucune connaissance a priori sur les sources.

Références

[*Burel*90] Gilles BUREL, Jean-Yves CATROS
"Réseaux de neurones en traitement d'images"
Bulletin d'information des Laboratoires Centraux de Thomson CSF,
n^o 4, décembre 1990

[*Grossberg*88] Stephen GROSSBERG
"The adaptative Brain"
North-Holland (Amsterdam), 1988

[*Hebb*49] D.O. HEBB
"The organization of behavior"
J. WILEY and Sons, 1949

[*LeCun*89] Yann LE CUN et al.
"Handwritten Digit Recognition :
Applications of Neural Network Chips and Automatic Learning"
IEEE Communications Magazine, November 1989

[*Rosenblatt*62] F. ROSENBLATT
"Principles of neurodynamics"
Spartan, New York, 1962

[*Rumelhart*86] D.E. RUMELHART and J.L. Mc CLELLAND
"Parallel Distributed Processing"
Bradford book - MIT Press - 1986

[*Saucier*89] G. SAUCIER, J. OUALI, J.L. PATRY, P. SILVESTRE
"Design of the VLSI demonstrator architecture"
Esprit Project n^o 2059 "Pygmalion", Report R211 (M12), December 1989

[*Sejnowski*87] T.J. SEJNOWSKI - C.R. ROSENBERG
"Parallel networks that learn to pronounce English text"
COMPLEX SYSTEMS 1 , pp145-168 , 1987

Chapitre 1:

DONNEES NEUROBIOLOGIQUES

Les astuces développées par l'évolution au cours des siècles peuvent être une source d'inspiration importante pour la mise au point de nouvelles méthodes de traitement du signal et de l'image. Notre objectif dans ce chapitre est de décrire quelques aspects des connaissances actuelles concernant le fonctionnement du système nerveux et l'organisation du traitement de l'information visuelle chez l'homme. Nous nous intéresserons plus particulièrement aux points qui semblent susceptibles de conduire à des modèles mathématiques exploitables pour la vision par ordinateur et le traitement du signal.

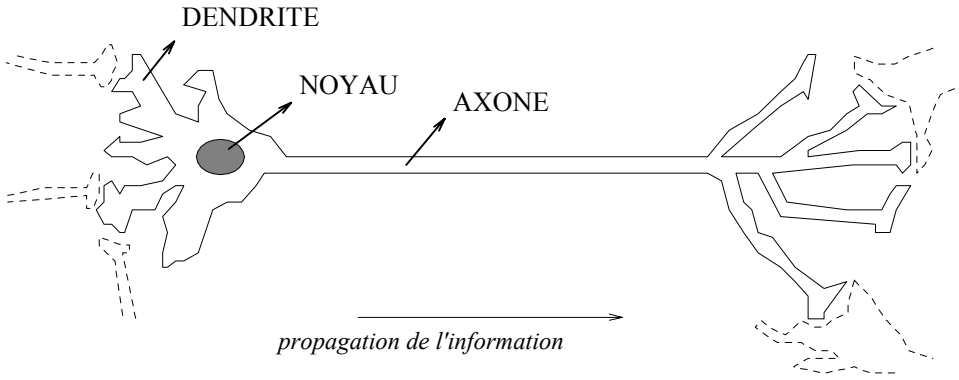
Dans une première partie, nous donnerons quelques indications concernant le fonctionnement de l'élément de base du tissu nerveux : le neurone. De ces indications nous dériverons un modèle mathématique qui pourra être implémenté dans un système artificiel.

Les connaissances actuelles concernant le traitement de l'image chez l'homme sont résumées dans une seconde partie. Nous tenterons d'en tirer quelques conclusions sur la façon dont l'information visuelle doit être traitée.

Enfin, une troisième partie traite d'un domaine encore très mal compris : l'apprentissage. La capacité d'apprentissage est une caractéristique fondamentale des systèmes vivants évolués, qui trouve sa source dans divers phénomènes biochimiques. Il est d'ailleurs certain que toute la circuiterie du système nerveux ne peut être prédéterminée génétiquement jusqu'au niveau du détail. En fait, c'est un ensemble restreint de lois particulièrement astucieuses qui permettent à cette circuiterie de s'adapter aux données provenant du monde extérieur.

1 Le neurone biologique

1.1 La cellule nerveuse



LE NEURONE BIOLOGIQUE

Figure 1:

Le type le plus commun de neurone biologique (fig 1) se présente comme une cellule dont le corps est doté de deux types de prolongements [$Taxi71$]:

- Des ramifications courtes et buissonnantes: les dendrites.

- Une longue fibre qui se termine par une arborisation: l'axone.

Ces prolongements permettent au neurone d'établir des connexions avec d'autres cellules nerveuses (ou avec des cellules musculaires dans le cas des neurones moteurs). L'élément le plus frappant à l'observation microscopique du tissu nerveux est la densité de connexions: chaque neurone communique, en moyenne, avec 10^4 autres neurones. C'est cette densité de connexion, cette importance des communications, qui fait toute la puissance du système nerveux. Le cerveau humain comporte de l'ordre de 10^{11} neurones, soit environ 10^{15} connexions.

Les dendrites constituent en quelque sorte les points d'entrée du neurone: c'est par les dendrites qu'il reçoit de l'information provenant d'autres neurones. Après traitement, l'information est propagée vers les neurones suivants, via l'axone. Les

terminaisons de l'axone forment des connexions (synapses) avec d'autres neurones.

Le fonctionnement du neurone met en jeu des phénomènes chimiques et électriques. Nous allons chercher, par une mise en équation simplifiée de ces phénomènes, à obtenir un modèle mathématique du neurone. Ce modèle sera une source d'inspiration intéressante pour la conception de nouveaux dispositifs de traitement du signal et de l'image.

1.2 L'influx nerveux

1.2.1 Le potentiel de repos

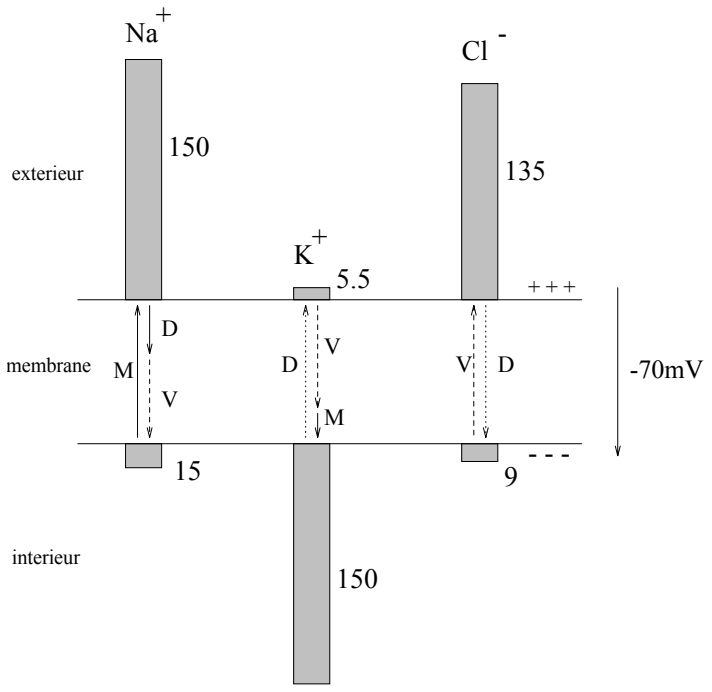


Figure 2:

La membrane du neurone est une membrane dialysante d'une épaisseur de l'ordre de $0.01\mu m$ [*Hermann*79]. Au repos, il existe une différence de potentiel de -70mV

entre l'intérieur et l'extérieur du neurone (potentiel de repos). Son existence est liée aux différences de concentrations ioniques de part et d'autre de la membrane (fig 2). A l'extérieur de la cellule, on trouve une forte concentration en sodium (Na^+) et en chlore (Cl^-). A l'intérieur, on trouve une importante concentration en potassium (K^+) et en anions organiques (non représentés sur la figure).

Les principaux mouvements ioniques mis en jeu dans le fonctionnement du neurone sont ceux du sodium et du potassium. La perméabilité de la membrane à ces ions dépend du potentiel. Du point de vue macroscopique, on peut distinguer trois flux correspondant à trois causes différentes:

1. La diffusion, qui tend à égaliser les concentrations (elle dépend du gradient de concentration et de la perméabilité de la membrane).
2. La force électrostatique, dûe à la différence de potentiel
3. La pompe métabolique.

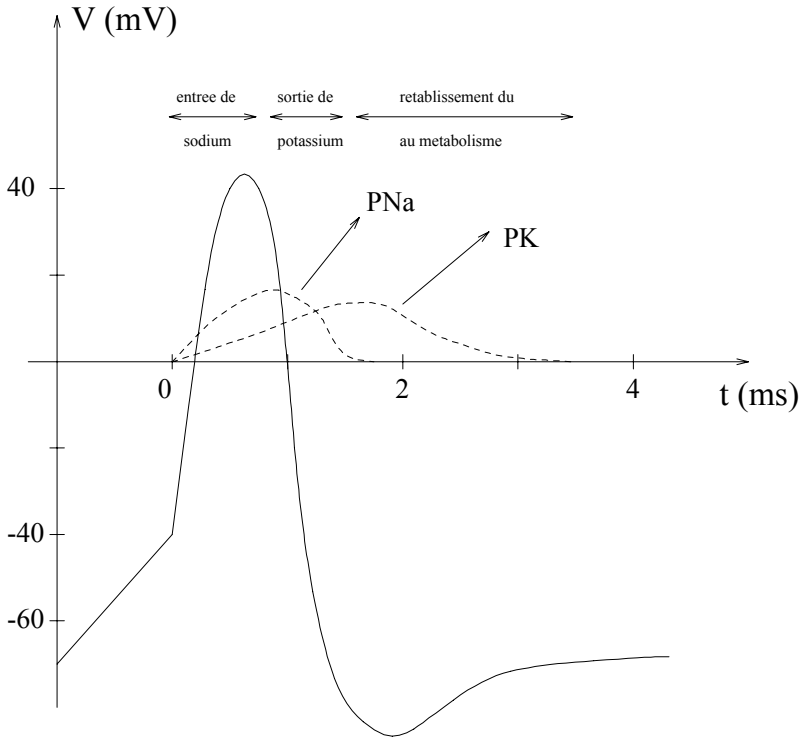
Au repos, ces trois flux s'équilibrent. La pompe métabolique est un mécanisme actif, consommateur d'énergie. En son absence, les concentrations tendraient à s'égaliser par le jeu de la diffusion.

1.2.2 Le potentiel d'action

Lorsque, à l'aide d'électrodes, on augmente artificiellement le potentiel de la membrane, il ne se passe rien jusqu'à une valeur critique de l'ordre de -40mV. Au dessus de cette valeur, on observe une inversion brutale du potentiel, puis un rapide retour à l'état de repos (fig 3), dans un temps de l'ordre de 3.5ms.

Cette inversion temporaire du potentiel, appelée potentiel d'action, a pour origine une avalanche de phénomènes électrochimiques [$Taxi71$] [$Hermann79$]. En effet, la perméabilité de la membrane dépend fortement du potentiel. Lorsque le potentiel dépasse -40mV, la perméabilité aux ions Na^+ augmente rapidement, d'où une entrée massive de ces ions dans la cellule. Ceci entraîne une augmentation du potentiel, d'où une perméabilité encore plus importante au sodium, etc.

L'inversion du potentiel résultant de cette réaction autocatalytique provoque une augmentation de la perméabilité aux ions K^+. Il semblerait en effet que la



POTENTIEL D'ACTION

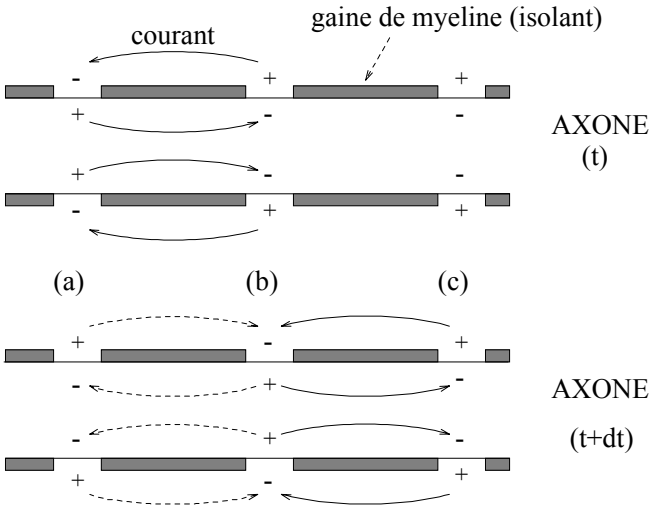
PNa : permeabilite au sodium
PK : permeabilite au potassium

Figure 3:

perméabilité aux ions K^+ augmente plus lentement, et avec un certain retard, par rapport à la perméabilité aux ions Na^+. Ceci induit une sortie massive d'ions potassium, d'où une chute du potentiel. Le potentiel de repos (-70mV) est ensuite rapidement rétabli par la pompe métabolique.

1.2.3 La propagation de l'influx nerveux

L'axone (ou fibre nerveuse) est en presque totalité recouvert d'une gaine de myéline qui joue le rôle d'un isolant [$Hermann79$]. Cette gaine est régulièrement interrompue. Lorqu'une inversion de potentiel (potentiel d'action) est créée au point (a) de l'axone (fig 4), il naît un courant vers le prochain point d'interruption de la gaine isolante (b). Ce courant fait monter le potentiel en (b), d'où le déclenchement d'un nouveau



PROPAGATION DE L'INFLUX NERVEUX

Figure 4:

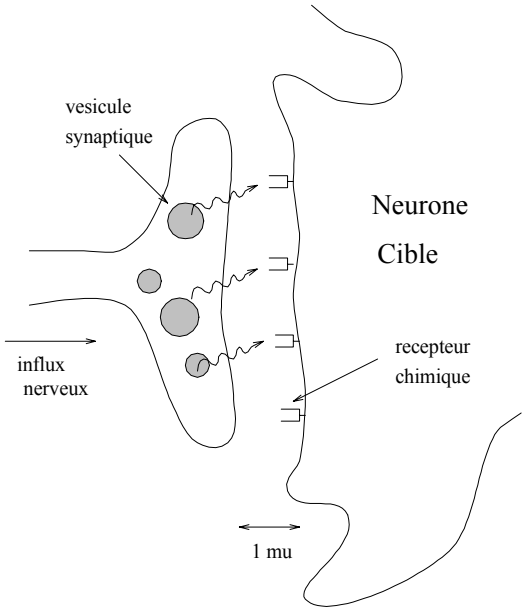
potentiel d'action. Pendant ce temps, l'équilibre se rétablit en (a).

Le même phénomène se produit ensuite du point (b) vers le point (c), et ainsi de suite. L'influx nerveux se propage donc comme une onde de dépolarisation. Il ne peut revenir en arrière car, au point (a), la fibre est dans un état réfractaire (non excitable) jusqu'au rétablissement des concentrations ioniques de repos par la pompe métabolique.

La vitesse de propagation dépend de la longueur L des zones isolantes, et de l'intensité I des courants locaux. En effet, un potentiel d'action se déclenchera en (b) lorsqu'une charge Q suffisante pour faire monter le potentiel au dessus de -40mV y sera parvenue, c'est à dire après un temps dt=Q/I. La vitesse de propagation de l'influx nerveux est donc v=LI/Q.

L'influx nerveux se propagera d'autant plus rapidement que l'axone est bon conducteur (donc en particulier de diamètre important), et que la gaine de myéline est peu interrompue. Par exemple, pour L=5mm et dt=1ms, la vitesse de propagation est de 5m/s. Expérimentalement on observe généralement des vitesses de 2 à 100 m/s.

1.3 La transmission synaptique



LA TRANSMISSION SYNAPTIQUE

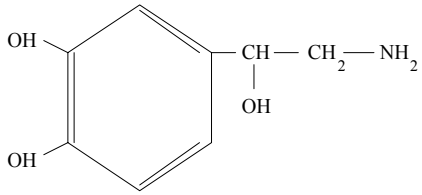
Figure 5:

Comment se fait la transmission de l'influx nerveux lorsqu'il arrive à une extrémité de l'axone (synapse)? Les expériences montrent que les cas de transmission électrique sont rares. Généralement, la transmission met en œuvre des processus chimiques par l'intermédiaire de molécules organiques: les neuromédiateurs [$Taxi71$] [$Hermann79$].

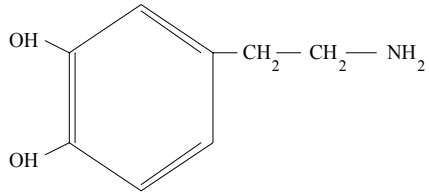
La terminaison synaptique contient des vésicules de neuromédiateurs (fig 5). L'arrivée de l'onde de dépolarisation provoque une augmentation de la perméabilité de la membrane aux neuromédiateurs. Ceux-ci se trouvent donc libérés dans l'espace intersynaptique (d'une épaisseur de l'ordre de $1\mu m$), et viennent se fixer sur des récepteurs chimiques situés sur le neurone cible. Lorsque ces récepteurs captent des neuromédiateurs, ils provoquent une augmentation de la perméabilité de la membrane du neurone cible aux ions sodium. L'entrée d'ions sodium dans la cellule peut ainsi déclencher un nouveau potentiel d'action suivant les principes précédemment énoncés (en pratique il faut souvent plusieurs dépolarisations successives pour libérer une quantité suffisante de neuromédiateur).

On peut citer comme neuromédiateurs la noradrénaline, la dopamine (fig 6), et l'acide gamma-aminobutyrique (GABA). Le GABA est présent dans les synapses inhibiteurs. Sa fixation sur les récepteurs a pour effet de bloquer provisoirement le

fonctionnement du neurone cible.



noradrenaline



dopamine

NEUROMEDIATEURS

Figure 6:

1.4 Modélisation du neurone

Macroscopiquement, on peut voir le potentiel d'action comme une impulsion. Le neurone transmet donc aux autres neurones un train d'impulsions. C'est la fréquence de ces impulsions qui est porteuse d'information, car leur amplitude est constante. Ce mode de codage en fréquence est connu pour être bien plus robuste que le codage en amplitude, c'est probablement pourquoi la sélection naturelle l'a choisi.

Considérons un neurone indicé j qui reçoit en entrée des trains d'impulsions de fréquences O_i (l'indice i parcourant les entrées). Notons W_{ij} le coefficient représentant la force de la connexion synaptique liant le neurone i au neurone j (sa valeur dépend notamment de la surface de connexion). Le problème que nous nous posons est de déterminer la fréquence O_j en fonction des O_i et des W_{ij}.

Pour cela, essayons d'estimer la quantité q_j de neuromédiateur fixé sur la membrane du neurone. Dans une synapse ij, on peut considérer que l'arrivée d'une impulsion libère une quantité élémentaire dq_{ij} de neuromédiateur, quantité qui est proportionelle à la force de la connexion. Si les impulsions arrivent avec une

fréquence O_i, on peut donc écrire:

$$\frac{dq_{ij}}{dt} = W_{ij}O_i$$

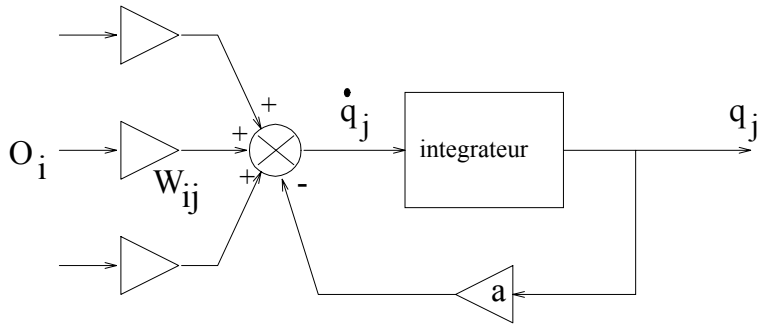
Un autre phénomène important est la destruction des neuromédiateurs par des enzymes (sans quoi la quantité de ces composés augmenterait indéfiniment). On peut supposer que ce phénomène vérifie la loi classique de dégradation chimique:

$$\frac{dq_j}{dt} = -\alpha q_j$$

Finalement, comme $dq_j = \sum_i dq_{ij}$, on a:

$$\frac{dq_j}{dt} = \sum_i W_{ij}O_i - \alpha q_j$$

Cette équation est illustrée sur la figure 7.



EVOLUTION DE LA QUANTITE DE NEUROMEDIATEUR

FIXE SUR LES RECEPTEURS

Figure 7:

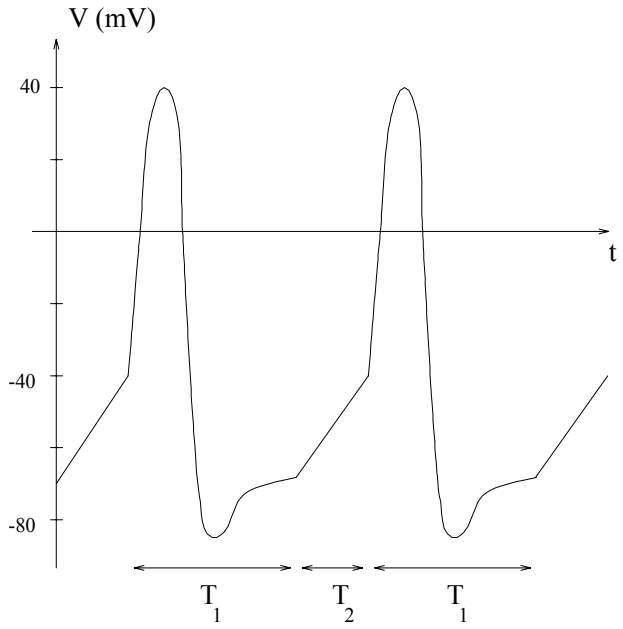
Posons à présent $X_j = \sum_i W_{ij}O_i$, et exprimons l'équation précédente en utilisant la transformée de Laplace:

$$\hat{q}_j = \frac{\hat{X}_j}{p + \alpha}$$

Si les fréquences d'entrée O_i varient suffisamment lentement (plus précisément, si elles varient peu sur une période de quelques α^{-1}), on peut faire une approximation quasi-statique, c'est à dire supposer qu'un état de quasi-équilibre est maintenu:

$$q_j = X_j/\alpha$$

La figure 8 représente l'évolution du potentiel du neurone. Entre deux impulsions, le potentiel augmente avec une vitesse proportionelle à la perméabilité aux ions Na^+, et donc à la quantité de neuromédiateur fixé sur les récepteurs.



TRAIN d'IMPULSIONS sur l'axone

Figure 8:

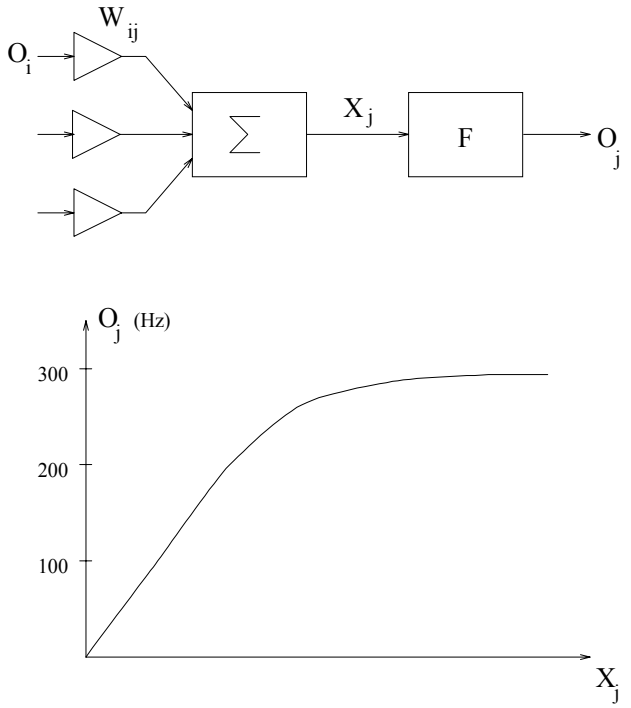
En notant β la constante de proportionalité, et $\Delta V = 30mV$, le temps nécessaire pour passer de -70mV à -40mV est:

$$T_2 = \frac{\Delta V}{\beta q_j}$$

En notant $T_1 = 3.5ms$ la durée du potentiel d'action et $k = \frac{\alpha}{\beta}\Delta V$, la fréquence du train d'impulsions est:

$$\mathbf{O_j = \frac{1}{T_1 + \frac{k}{x_j}}}$$

L'allure de cette courbe , que nous nommerons "fonction de transition du neurone" est représentée figure 9, ainsi que le modèle mathématique obtenu pour le neurone. Notons que la fonction de transition réelle est probablement plus complexe car nous avons fait des hypothèses de proportionalité qui ne sont sans doute pas respectées.



LE MODELE DU NEURONE BIOLOGIQUE

Figure 9:

Toutefois, l'allure générale de la courbe réelle est probablement la même, c'est à dire qu'elle comprend:

- Une zone quasi-linéaire près de l'origine.
- Une zone de saturation aux environs de 300Hz.

En résumé, nous avons établi un modèle simplifié du neurone qui fait ressortir deux blocs fonctionnels importants:

- Un sommateur pondéré
- Une non-linéarité présentant une zone quasi-linéaire et une zone de saturation.

Les neurobiologistes ont déjà abouti à des conclusions similaires depuis plusieurs dizaines d'années par des raisonnements qualitatifs. L'élément le plus frappant est la limitation de la fréquence de sortie à 300Hz. Du point de vue de l'électronicien, le neurone biologique est donc un composant extrêmement lent (on a l'habitude des

microprocesseurs dont l'horloge tourne à quelques dizaines de Mega-Hertz). Les performances du cerveau humain ne peuvent par conséquent s'expliquer que par la densité des connexions, ainsi que par le fait que l'information est traitée en parallèle par quelques 10^{11} neurones, et non pas par les performances du composant de base.

2 Le système visuel humain

2.1 Généralités

Il est bien connu que des problèmes aisés à résoudre pour l'être humain (par exemple localiser et reconnaître les divers objets présents sur une image, ou encore comprendre l'écriture manuscrite) sont très difficiles pour l'ordinateur. On peut donc émettre l'hypothèse que le cerveau humain est plus performant parce qu'il utilise un modèle de traitement de l'information très différent de ceux qui ont cours aujourd'hui en informatique. Ainsi, le cerveau traite l'information de façon massivement parallèle. La mémoire y est diffuse et non pas localisée comme dans un ordinateur classique, ce qui le rend très robuste par rapport aux dégradations (on perd quelques centaines de milliers de neurones par jour sans dégradation significative des performances). De plus, la notion de programmation n'existe pas: l'acquisition des connaissances se fait par apprentissage, c'est à dire par organisation automatique de la structure interne en fonction des données reçues de l'extérieur.

Cette section a pour objet de décrire l'organisation du système visuel humain telle qu'elle est connue actuellement. Nous discuterons ensuite des implications éventuelles de ces connaissances sur la conception de systèmes artificiels de traitement d'images.

Les mécanismes physiologiques de la vision ont pour point de départ la formation d'images sur la rétine. Cette image est une projection bidimensionnelle du monde extérieur, suivant les lois de l'optique géométrique. A l'état brut (c'est à dire au niveau de cellules photoréceptrices), cette image n'est qu'une mosaïque de points lumineux sans signification particulière. Le cerveau doit donc effectuer un traitement élaboré pour extraire de cette mosaïque de points les informations qui ont un sens pour lui, c'est à dire qui lui permettent de se forger une représentation du monde extérieur, et autorisent des déplacements "intelligents" dans l'espace (saisie d'un objet, évitement d'obstacles, etc).

Par exemple, une information particulièrement importante à extraire d'une image est la nature des objets qui y sont présents et leur position spatiale. Contrairement aux

apparences, il s'agit d'une tâche extrêmement difficile. Aucun système artificiel n'est actuellement capable de la réaliser, sauf dans des cas très favorables (nombre restreint d'objets bien connus et correctement éclairés sur fond uniforme par exemple). En effet, l'identification d'objets dans le monde tridimensionnel soulève des problèmes d'une grande complexité :

1. Il existe de très fortes variations à l'intérieur d'une même classe d'objets. Ceci est encore plus vrai pour les objets non rigides.
2. Un objet peut être vu sous un angle quelconque.
3. Un objet peut être partiellement occulté.
4. Les conditions d'éclairage ne sont jamais constantes. De plus des ombres et des reflets peuvent être présents.

Malgré ces problèmes, nous sommes capables d'identifier les divers objets d'une image avec une remarquable efficacité. Ceci suppose les caractéristiques suivantes pour le système visuel humain:

1. Il doit disposer de modèles des divers objets susceptibles d'apparaître sur une image, et être capable de comprendre la scène, sans quoi il lui serait impossible d'isoler les objets du fond sauf dans des cas très simples (objets de couleur ou de luminance différente du fond). Un objet présente en général sur sa surface des points de couleurs et de luminances différentes. Or nous sommes bien capables de regrouper ces points d'images comme appartenant à une même entité, ce qui confirme la nécessité d'un modèle a priori.
2. Ces modèles doivent pourtant être extrêmement souples car nous sommes capables de reconnaître un objet quel que soit l'aspect sous lequel il se présente (orientation, éclairage, occlusion partielle, déformation pour les objets non rigides, ...).

2.2 La rétine

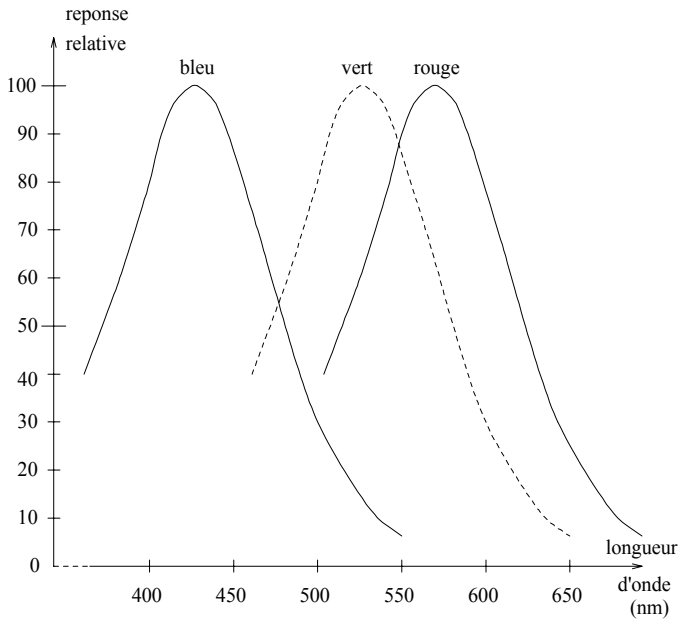
La rétine est un tissu nerveux tapissant le fond de l'oeil. Elle comprend une couche de cellules photoréceptrices et 2 couches de neurones [*Imbert*83].

La couche de cellules photoréceptrices constitue une mosaïque d'environ 160 millions de points. Chacune de ces cellules transforme l'intensité lumineuse qu'elle

reçoit en une tension électrique plus ou moins proportionnelle. On distingue 2 types de cellules: Les bâtonnets (150 millions) qui sont sensibles à l'intensité lumineuse sur la totalité du spectre visible, et les cônes (7 millions) qui réagissent à l'intensité dans une bande de fréquence particulière [$Nathans89$]. Les cônes contiennent en effet une protéine filtrante (ou pigment). Trois types de cônes existent, et sont classés en fonction du spectre d'absorption de leur pigment (rouge, vert, bleu) (fig 10). Les cônes sont particulièrement concentrés dans la fovea (zone centrale de la rétine). On pourrait penser que les bâtonnets sont inutiles car il est possible de reconstituer la luminance à partir des trois canaux (rouge, vert ,bleu). En fait leur intérêt est double:

1. Ils permettent une résolution plus fine (il suffit d'un bâtonnet par point d'image alors que trois cônes sont nécessaires).

2. Les cônes ne dispensent plus de signal exploitable dès que l'éclairage est faible du fait de la perte d'énergie dûe à leur pigment (même dans leur bande de fréquence privilégiée). On peut vérifier aisément ce phénomène en constatant qu'il nous est impossible de distinguer les couleurs lorsque l'éclairage est faible. Les bâtonnets sont donc particulièrement utiles pour la vision nocturne.



REPONSE SPECTRALE DES CONES

Figure 10:

Les cellules photoréceptrices transmettent l'information électrique (tension) aux neurones de la première couche (neurones bipolaires). Ces neurones ne semblent

pas effectuer de traitement très élaboré si ce n'est un transcodage de la tension électrique en fréquence d'impulsions (influx nerveux). Mais la découverte récente d'interactions latérales via des cellules non-neuronales [*Piccolino*88] laisse supposer qu'ils ont peut-être un rôle dans l'adaptation du système visuel au contraste et à la luminosité de l'image (ils réaliseraient en quelque sorte une normalisation).

Les neurones de la deuxième couche (neurones ganglionnaires) reçoivent les influx nerveux provenant des neurones bipolaires. Depuis les années 50, il est possible de déterminer le stimulus visuel auquel répond une cellule quelconque. Pour cela, on place une micro-électrode près de l'axone de cette cellule, et on recherche la stimulation visuelle qui provoquera la plus forte réponse (fréquence maximale du train d'impulsions). Ainsi, on a pu montrer que les neurones répondent à des stimulus de plus en plus élaborés à mesure que l'on se déplace vers des zones cervicales éloignées de la rétine. Dans les zones proches de la rétine, les neurones répondent à des formes simples (lignes, cercles,...) alors que dans les zones éloignées ils répondent uniquement à des stimulus complexes (visage, objet familier, ...).

Cette technique a permis de montrer que chaque neurone ganglionnaire a un champ récepteur, c'est à dire qu'il ne répond qu'aux stimulations situées dans la zone de la rétine qui lui est proche. Ceci est aisément explicable par le fait que les connexions sont locales. De plus, ces neurones sont sensibles non pas à l'éclairage moyen dans leur champ récepteur, mais à la différence d'éclairage [*Imbert*83] entre un disque central et un anneau périphérique (fig 11). Ce type de réaction est probablement le résultat de la répartition spatiale des connexions excitatrices et des connexions inhibitrices. Il existe 2 types de neurones : les neurones à centre "ON" et les neurones à centre "OFF" dont les réponses sont antagonistes. Leur réponse impulsionnelle (dans le domaine spatial) peut être modélisée assez précisément (fig 11) par une différence de gaussiennes [*Watson*89]: la fonction DOG (Difference Of Gaussians).

Chaque grande classe (ON/OFF) peut à son tour être divisée en deux sous classes: Les neurones X dont l'activité spontanée est nulle et qui répondent de façon prolongée à un stimulus, et les neurones Y qui ont une faible activité spontanée et qui ne répondent que de façon transitoire à un stimulus. Les neurones Y semblent donc particulièrement adaptés à la détection de mouvement. Ils sont d'ailleurs plus nombreux dans les régions périphériques de la rétine, régions qui sont connues pour être très sensibles aux objets en mouvements, alors que la partie centrale est plutôt impliquée dans l'analyse fine d'images fixes. Les neurones Y ont également un champ récepteur plus étendu que les neurones X.

On peut se demander quel est l'intérêt de la présence de 2 couches de neurones dans une zone où ils sont a priori bien plus vulnérables qu'à l'intérieur de la boîte crânienne.

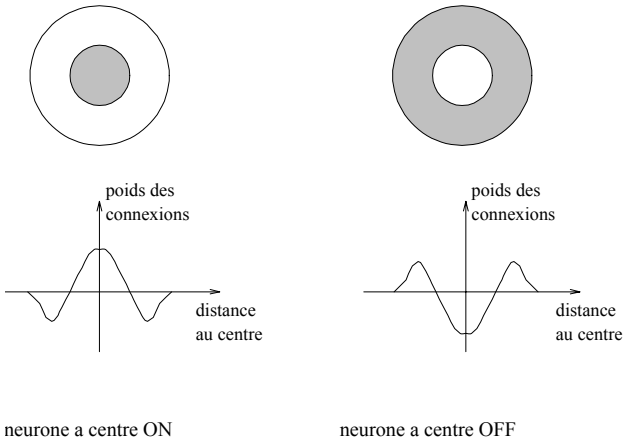


Figure 11:

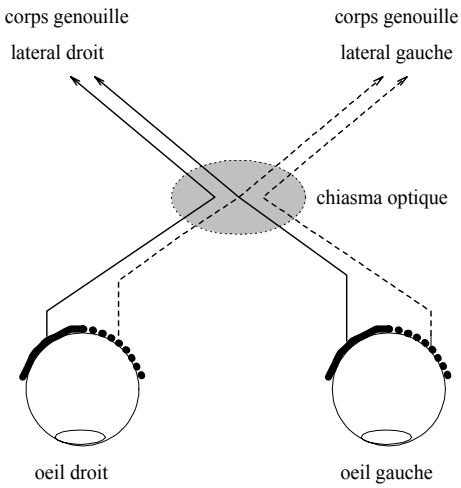
La première couche (neurones bipolaires) est justifiée par le fait qu'il semble difficile de transmettre directement les tensions de sortie des cellules photoréceptrices jusqu'au cerveau. Sous cette forme (codage en amplitude), l'information serait trop vulnérable aux perturbations diverses, et arriverait extrêmement bruitée à destination. Le codage en fréquence réalisé par les neurones bipolaires est connu pour être bien plus robuste, et peut donc être transmis sans dommages sur de longues distances. La deuxième couche a pour objet de réaliser une compression de l'information afin de réduire le diamètre du nerf optique à une taille acceptable.

Les axones des neurones ganglionnaires constituent le nerf optique. Ils sont au nombre de 1 million, valeur qui est à comparer aux quelques 160 millions de cellules photoréceptrices. La compression de données réalisée dans la rétine est donc très importante. Il est probable que cette compression s'effectue sans perte importante d'information significative. En effet, le nombre de "canaux" a été réduit d'un facteur supérieur à 100, mais d'un autre coté ce sont des informations de plus haut niveau qui transitent par le nerf optique. Ainsi, l'information transmise correspond aux discontinuités et au mouvement, et laisse tomber tout ce qui est uniforme ou statique. De plus, il est possible que certains neurones ganglionnaires réalisent un multiplexage temporel de l'information, en transmettant préférentiellement le mouvement ou les discontinuités de luminance suivant les circonstances.

Notons que chez certains animaux primitifs la compression est encore bien plus importante que chez l'homme. Ainsi, chez la grenouille, certains neurones

ganglionnaires réagissent de façon très sélective à la présence d'un petit objet sombre animé d'un rapide mouvement de va et vient dans son champ récepteur [*Imbert*83]. Si l'on excite artificiellement un tel neurone, la grenouille tente de gober une proie imaginaire. Il s'agit donc en quelque sorte d'un neurone "détecteur de mouche". En conséquence, ce sont des informations particulièrement significatives pour l'animal qui vont transiter par son nerf optique, ce qui compense la réduction du nombre de fibres.

2.3 Un relais intra-cérébral : le corps genouillé latéral

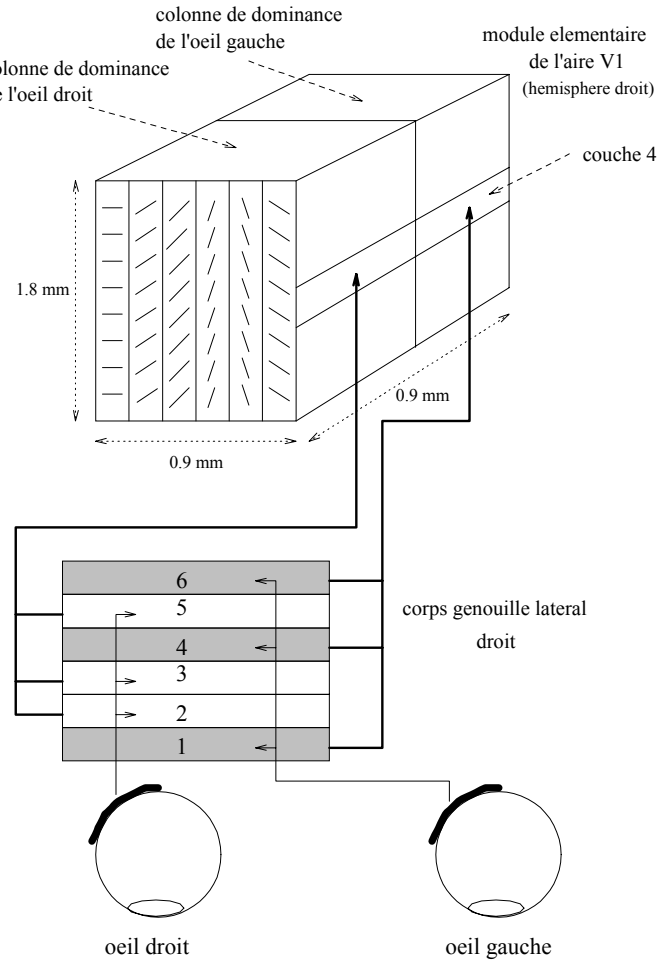


CHIASMA OPTIOUE

Figure 12:

Les axones des neurones ganglionnaires se croisent dans le chiasma optique (fig 12), de telle sorte que la moitié gauche du champ visuel est analysée par l'hémisphère droit, et inversement. Puis ils se prolongent vers 2 groupements de neurones: les corps genouillés latéraux, qui jouent le rôle de relais intra-cérébraux. Chacune de ces structures est constituée de 6 couches de neurones [*Imbert*83]. Les couches 1,4,6 reçoivent les fibres provenant de l'oeil situé du côté opposé, et les couches 2,3,5 celles provenant de l'oeil situé du même côté. Ainsi, un point situé dans la moitié gauche du champ de vision (fig 13) sera représenté dans le corps genouillé droit (couches 1,4,6 pour les fibres venant de l'oeil gauche, et couches 2,3,5 pour celles venant de l'oeil droit).

Il semble que les corps genouillés n'effectuent pas de traitement particulier de



CIRCUITS VISUELS PRIMAIRES

Figure 13:

l'information, et jouent simplement le rôle d'un relais. En effet, D. Hubel et T. Wiesel montrèrent dès 1961 (rapporté dans [*Imbert*83]) que les neurones des corps genouillés réagissent aux mêmes stimulus que les neurones ganglionnaires de la rétine. Par contre, ils semblent y réagir avec plus de sensibilité.

2.4 Le cortex visuel primaire

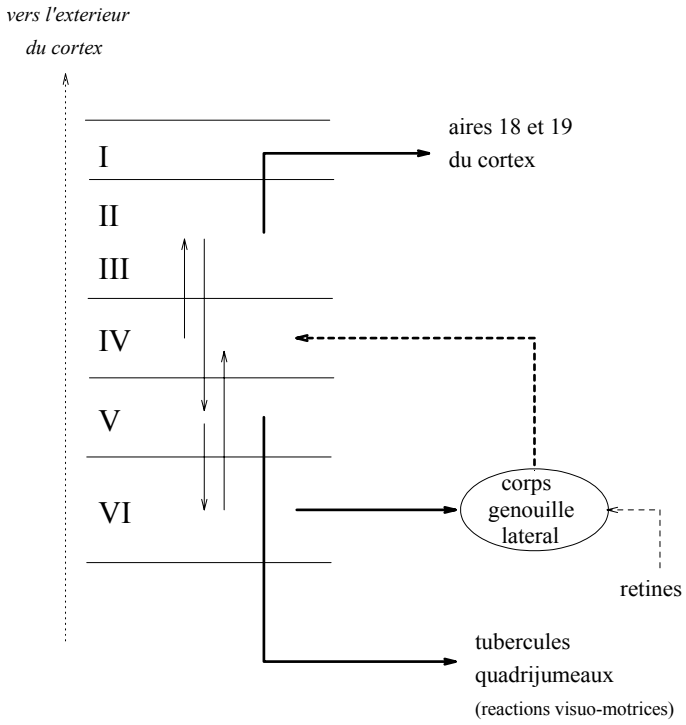
Les axones issus des corps genouillés latéraux traversent la substance blanche et viennent établir des connexions avec les neurones de l'aire 17 du cortex cérébral (selon la classification de Bormann [1905], qui a délimité et numéroté 40 aires différentes

dans le cortex cérébral). L'aire 17 est située dans la partie la plus postérieure de cerveau (cortex occipital).

On rappelle que le cortex cérébral (ou matière grise) correspond à la partie extérieure du cerveau, et est le siège des traitements élaborés de l'information, les traitements primaires étant laissés à des structures plus anciennes telles que le cervelet. Le cortex est une structure qui est apparue assez tardivement au cours de l'évolution. Son épaisseur varie de 1.5 à 5 cm, et, compte tenu des diverses circonvolutions, sa surface atteint 2000 cm^2. Il est constitué d'une superposition de plusieurs couches, et contient de l'ordre de 80000 neurones sous chaque unité de surface de $1mm^2$, sauf dans l'aire 17 où ce chiffre atteint 200000 [$Crick$86]. On distingue en général 6 couches suivant la forme des neurones présents. La couche la plus externe ne contient pas de neurones: elle est formée d'axones et constitue en quelque sorte un bus de données. En général les informations sensorielles arrivent au cortex par l'intermédiaire d'une structure centrale: le Thalamus. Les corps genouillés latéraux font partie du Thalamus.

L'aire 17 de la substance grise, encore appelée aire striée ou aire visuelle primaire (V1) a une épaisseur de 1.8cm. Les fibres issues des corps genouillés se terminent principalement dans la couche 4. Les connexions internes à l'aire V1 et ses prolongements vers l'extérieur sont schématisés figure 14 (notons que ce type d'organisation se retrouve dans l'ensemble du cortex [$Crick$86]). La couche 4 connecte les couches 2 et 3, qui à leur tour connectent la couche 5. La couche 5 émet ses prolongements vers la couche 6, qui reboucle, via une branche colatérale ascendante, vers la couche 4. Les connexions avec l'extérieur sont assurées par les couches 2 et 3 qui émettent vers les aires 18 et 19 du cortex visuel. La couche 5 émet vers les tubercules quadrijumeaux antérieurs (ces structures jouent un rôle essentiel dans l'élaboration des réactions visuo-motrices et pourraient donc déclencher la capture d'une cible intéressante par le regard). Quant à la couche 6, elle reboucle vers les corps genouillés latéraux. L'effet de ces rebouclages, qui se retrouvent également dans les aires supérieures du cortex visuel, est mal connu. Ils pourraient avoir un rôle dans la prise en compte du contexte, la stabilisation du système global, ou encore dans l'apprentissage (renforcement).

On a montré expérimentalement que les neurones de l'aire V1 ont des champs récepteurs allongés de forme elliptique[$Imbert$83], et se comportent généralement comme des détecteurs de contours. Chaque neurone réagit donc à une transition lumineuse (contour) d'orientation particulière. On distingue les cellules simples qui ne répondent que si le contour a une position bien précise dans le champ récepteur, et les cellules complexes qui répondent quelle que soit la position du contour dans le champ récepteur (à condition toutefois que son orientation soit correcte). Les cellules simples se trouvent surtout dans la couche 4 (ce sont donc les plus proches



ENTREES/SORTIES de L'AIRE 17 (V1)

Figure 14:

de la rétine en terme de distance de connexion), et les cellules complexes se trouvent surtout dans les autres couches.

L'analyse détaillée de la structure de l'aire 17 montre qu'elle est constituée d'un pavage de modules élémentaires tel que représenté figure 13. Ce module de base est formé de colonnes de dominance oculaire. Dans chaque colonne les neurones réagissent préférentiellement aux informations provenant de l'oeil correspondant. Ces colonnes sont elles mêmes formées de colonnes d'orientation dominante dans lesquelles les neurones répondent préférentiellement à un contour d'orientation donnée. Chaque colonne d'orientation a une largeur de $50\mu m$. Entre 2 colonnes successives, l'orientation dominante varie d'environ 10^o. Il y a environ 18 colonnes d'orientation afin de décrire les 180^o [$Imbert83$]. Ce module élémentaire est répété un grand nombre de fois afin d'analyser tout le champ de vision.

Pour terminer, signalons que des modèles mathématiques plus précis de l'organisation du cortex visuel primaire ont été proposés ([$Watson89$] [$Porat89$] [$Daugman89$] par exemple). Toutefois, il convient de noter que ces modèles ont plus été établis sur des contraintes de traitement du signal que sur le résultat d'expérimentations biologiques. Les contraintes imposées (principalement une contrainte de représentation compacte

de l'information) ne semblent pas vraiment justifiées. En effet, une représentation compacte de l'information dans l'aire V1 peut être nuisible, car sous cette forme elle peut être difficile à exploiter par les aires supérieures. Il serait probablement plus efficace de tolérer une certaine redondance de l'information. On devra certes allouer un plus grand nombre de neurones à l'aire V1, mais les aires supérieures verront leur tâche facilitée. Les expériences biologiques semblent d'ailleurs confirmer une représentation redondante de l'information dans V1 [*Daugman*89].

2.5 Le cortex associatif

Les neurones des couches 2 et 3 de l'aire V1 émettent leurs axones vers des aires visuelles voisines : L'aire 18 de Bormann (qui comprend les aires visuelles V2,V3,V4) et l'aire 19 (qui comprend l'aire visuelle V5). Ces dernières émettent à leur tour des prolongements vers les aires 20 et 21 du cortex temporal, ainsi que vers l'aire 7 du cortex pariétal [*Imbert*83].

Il semble que les aires V2 à V5 soient relativement spécialisées. Ainsi, V5 serait spécialisée dans l'analyse du mouvement, V4 dans le traitement de la couleur, et V3 dans l'analyse des formes. L'aire V2 est moins spécialisée, mais contient un grand nombre de neurones binoculaires précisément accordés sur une distance (perception de la profondeur par stéréoscopie).
Cette spécialisation peut par exemple être mise en évidence grâce à la tomographie par émission de positons (particules d'anti-matière correspondant aux électrons), qui permet de mesurer l'activité des zones cérébrales [*Zeki*90]. Ainsi, lorsqu'un sujet regarde une image fortement colorée, on détecte une forte augmentation de l'activité métabolique dans l'aire V4. Toutefois, la spécialisation n'est pas complète car dans chaque aire on trouve des neurones répondant à des stimulus non dominants. De plus, il existe une importante circulation d'informations entre les diverses aires visuelles.

Les aires V2 à V5 émettent des prolongements vers le cortex temporal et pariétal. Le cortex temporal est spécialisé dans l'identification d'objets, et le cortex pariétal dans la reconnaissance de la configuration spatiale.

La structure de ces aires est assez mal connue. On sait simplement que les neurones de ces régions réagissent à des stimulus de plus haut niveau que les neurones de l'aire V1. Ainsi, on a trouvé dans le cortex temporal des neurones qui s'activent uniquement lorsqu'un visage apparait dans le champ de vision [*Thorpe*88], ou encore des neurones qui réagissent lorsque de la nourriture est aperçue. Ces neurones sont

généralement peu sensibles à la position particulière du stimulus dans le champ de vision. On a donc perdu en localisation au profit d'une abstraction plus grande.

2.6 Les performances du système visuel

2.6.1 Résultats expérimentaux

Il est intéressant d'avoir une idée des performances du système visuel en terme de rapidité d'analyse d'une image. Diverses études expérimentales ont été réalisées au cours des décennies précédentes. Le principe le plus simple consiste à présenter des images d'objets à des sujets et à mesurer le temps qu'il faut pour les nommer. En moyenne on trouve des valeurs de l'ordre de 750ms (la fourchette va de 600 ms pour des objets bien connus, à 1200 ms pour des objets moins familiers). Le problème est que l'on mesure non seulement le temps d'analyse de l'image par le système visuel, mais aussi le temps nécessaire à l'élaboration et la prononciation du nom. Mais on a au moins un majorant du temps nécessaire au système visuel pour identifier un objet.

Des expériences récentes plus élaborées menées par Potter au MIT puis par Thorpe à l'institut de neurosciences de Paris VI, tendent à éliminer la durée nécessaire à l'expression verbale [*Thorpe*88]. Sans décrire en détail le protocole expérimental utilisé par Thorpe disons simplement que l'idée consiste à présenter une succession de 4 images au sujet, puis à lui demander de nommer les 4 objets aperçus. Ainsi, pour une durée de présentation de chaque image de 60ms, on a 70% d'identifications correctes. A 140ms on atteint 84% d'identifications correctes. Ce pourcentage n'augmente quasiment plus si l'on prolonge la présentation (85% pour 200ms de présentation). On peut donc en conclure que 140ms est la durée nécessaire au système visuel pour identifier un objet. Une objection que l'on pourrait faire à cette expérience est que le cerveau continue peut être à analyser l'image après sa présentation en utilisant la mémoire. Selon Thorpe, ceci est peu probable car il serait alors perturbé par l'arrivé de la nouvelle image à analyser, mais il convient toutefois d'être prudent quant à l'interprétation de ces expériences.

Cependant, ce résultat semble confirmé par des expériences neurobiologiques. Ainsi on a trouvé dans le cerveau du singe une population de neurones qui s'activent uniquement lorsque l'animal voit de la nourriture. Des mesures par implantation de micro-électrodes ont montré que ces neurones commencent à s'activer 150ms après l'apparition d'un objet comestible dans le champ de vision (banane, cacahouète,etc).

Combien d'objets différents est-on capable d'identifier? Thorpe a abouti à

une estimation de l'ordre de 60000 en comptabilisant les objets simples (chaise, marteau, autruche, ...), les objets célèbres (La Joconde, La tour Eiffel, ...), et en estimant le nombre d'objets familiers (visages et lieus familiers, ...).

2.6.2 Conséquences

Quelles conclusions peut-on tirer des résultats précédents? D'abord, en ce qui concerne la rapidité d'analyse, les mesures citées ont des conséquences extrêmement importantes. Le signal provenant de la rétine doit traverser au moins 8 couches de neurones avant de parvenir dans les aires du cortex où les neurones répondent à des stimulus suffisamment complexes (cortex temporal pour l'identification d'objets): 2 couches dans la rétine, une couche dans les corps genouillés, 2 couches dans l'aire V1, probablement au moins 2 couches dans l'une des aires V2 à V5, et au moins une couche dans le cortex temporal.

Combinons les trois connaissances suivantes:

1. La fréquence de sortie du neurone est limitée à quelques 300Hz.
2. L'analyse d'une image peut être faite en 140ms.
3. Le signal provenant de la rétine doit traverser au moins 8 couches de neurones avant de parvenir dans les aires du cortex où les neurones répondent à des stimulus suffisamment complexes (cortex temporal).

Le nombre de potentiels d'actions autorisés par neurone est donc en moyenne $(300\text{x}0.14)/8 = 5$. On peut en tirer les conclusions suivantes:

1. Les neurones fonctionnent quasiment en binaire, car avec un maximum de 5 potentiels d'action autorisés, on peut tout au plus coder 5 niveaux différents.
2. Les rebouclages ne sont probablement pas exploités du fait du manque de temps. Ils jouent sans doute un rôle surtout quand l'image à analyser est "difficile" (faible contraste, scène très complexe, occlusions partielles, etc).

Pour être plus précis quant au premier point, il est probable que les neurone proches de la rétine (jusquà l'aire V1 incluse) doivent être capable de représenter le signal

avec un minimum de précision et ne peuvent donc pas fonctionner en binaire. Si on leur accorde 10 potentiels d'action afin de représenter une dizaine de niveaux, on doit admettre que les neurones des aires supérieures du cortex fonctionnent en binaire afin de ne pas contredire les mesures expérimentales.

Les remarques précédentes montrent que le cerveau n'utilise pas de méthodes itératives complexes pour traiter une image. Ce type de méthode est en plein développement dans le domaine du traitement d'image (méthodes de relaxation, champs Markoviens, etc). Ces méthodes, qui présentent l'avantage de reposer sur une construction mathématique relativement solide, ont donc probablement peu de vraisemblance biologique.

Le cerveau identifie en général un objet en un seul passage à travers un dizaine de couches de neurones. Il réserve l'exploitation des rebouclages aux cas difficiles (mauvaises conditions d'éclairage ou scènes très complexes). Dans ce cas, la durée d'analyse est obligatoirement plus importante. Mais même dans les cas difficiles, la durée nécessaire à l'identification d'un objet dépasse très rarement la seconde, ce qui montre que le nombre d'itérations est de toute façon très réduit. Si l'objet n'a pas été identifié après quelques secondes, en général il ne le sera pas du tout, même avec une observation plus prolongée.

Pour conclure, disons que les connaissances actuelles conçernant le système visuel humain montrent qu'il est possible d'identifier un objet parmi quelques 60000 possibles avec au plus une dizaine d'étapes de calcul massivement parallèle (la dizaine de couches de neurones).

3 L'apprentissage

3.1 Généralités

L'apprentissage est le mécanisme par lequel le système nerveux s'adapte au monde extérieur. Cette adaptation est réalisée principalement par une modification des connexions synaptiques. Malheureusement, les lois régissant ces modifications sont très mal connues. Nous donnons ci-dessous quelques hypothèses qui sont probablement très incomplètes.

On peut resituer l'apprentissage dans le cadre de la question plus générale suivante: "Comment la circuiterie du système nerveux est elle mise en place, et dans

quelle mesure est elle prédéterminée génétiquement?"

Pour tenter de répondre à cette question, il convient de distinguer 3 étapes importantes dans le développement du cerveau humain :

1. La mise en place globale des liaisons nerveuses.
2. L'établissement précis des connexions entre neurones.
3. L'ajustement de la force de chaque connexion.

Pour chacune de ces phases, nous allons tenter de déterminer la part de ce qui relève du déterminisme génétique, et la part de ce qui est le fruit de l'expérience, par le fait de l'apprentissage.

3.2 Mise en place des fibres nerveuses

Toutes les expériences biologiques confirment le fait que les grandes voies de communication entre les différentes régions du cerveau sont mises en place dès la naissance, et ne sont donc en aucune manière le fruit de l'expérience. Le déterminisme génétique est ici total. Mais on peut par exemple se demander comment, lors de leur croissance, les axones des neurones ganglionnaires de la rétine savent qu'ils doivent se diriger vers les corps genouillés latéraux et non pas vers toute autre aire cervicale. Ceci pourrait être explicable par deux phénomènes [*Kennedy*87]. D'abord, une "colle" chimique maintient groupés les axones provenant d'une même région, ce qui maintient l'organisation topologique. Ensuite, il semblerait que la zone cible diffuse un marqueur chimique spécifique, dont les axones suivent le gradient de concentration pendant leur croissance.

Notons qu'une meilleure compréhension de ces phénomènes pourrait peut-être permettre de mettre au point une technique visant à déclencher chimiquement un processus de réparation automatique de tissus nerveux endommagés.

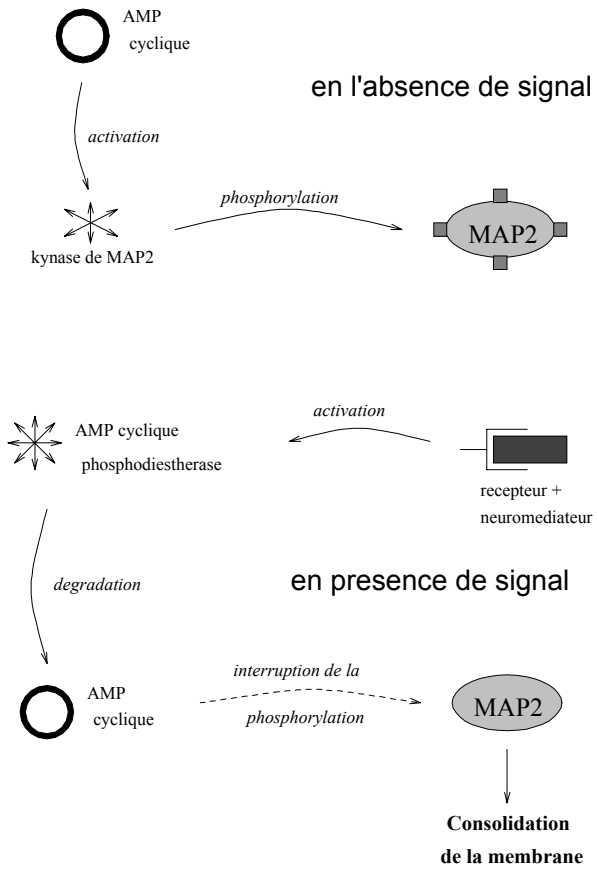
3.3 Etablissement des connexions

L'établissement des connexions entre neurones continue à se poursuivre plusieurs mois après la naissance, et semble jouer un grand rôle dans l'adaptation du cerveau au monde extérieur.

Une théorie qui semble confirmée par l'expérience a été proposée par Aoki et Siekevitz [*Aoki*88]. Une protéine, MAP2, intervenant dans la rigidité de la membrane neuronale serait l'élément déterminant impliqué dans la plasticité du cerveau. Il s'agit d'une énorme molécule, de masse moléculaire égale à 300000. A l'état normal, elle joue le rôle d'un "ciment" qui consolide les briques de base de la membrane : les micro-tubules (formés d'une autre protéine: la tubuline) et les micro-filaments. Mais, en fait, dès sa synthèse, des groupements phosphate viennent se fixer sur la molécule de MAP2 et modifient sa structure tridimensionnelle. Elle devient alors incapable de jouer son rôle de consolidation, et la membrane neuronale reste non-rigide. Cette phosphorylation est provoquée par une kynase (enzyme) activée par l'AMP cyclique (Adénosine MonoPhosphate cyclique) présent normalement dans la cellule (fig 15).

Ainsi, les dendrites sont initialement peu-rigides et se déplacent donc aléatoirement dans l'espace environnant le neurone. Au cours de ses déplacements aléatoires, une dendrite rencontre des terminaisons d'axone. Si l'axone rencontré est peu actif, il ne se passe rien et la dendrite reste non-rigide. Dès qu'elle rencontre un axone actif, il se produit une avalanche de phénomènes chimiques qui vont aboutir à la rigidification de la dendrite. En effet, si l'axone est actif, des neuromédiateurs vont venir se fixer sur les récepteurs de la membrane dendritique. L'association neuromédiateur-récepteur catalyse la synthèse de l'AMP cyclique phosphodiesthérase, un enzyme qui dégrade l'AMP cyclique . Donc la kynase responsable de la phosphorylation de MAP2 n'est plus activée. Par conséquent, MAP2 peut jouer son rôle dans la consolidation de la membrane. Ce phénomène est en général irréversible, ce qui explique le fait qu'il intervient peu chez l'adulte.

Cette théorie permet de répondre à une question importante concernant le système visuel humain: est il entièrement déterminé génétiquement ou est il partiellement bâti par l'expérience? Des expériences ont été réalisées sur des chats, dont l'organisation cérébrale est proche de celle de l'homme. Ainsi, on n'a mesuré aucune modification de la phosphorylation des molécules MAP2 dans la rétine et le corps genouillé durant les mois suivant la naissance. Par contre, des modifications importantes ont été détectées dans l'aire V1. On peut donc en conclure que les connexions des neurones du corps genouillé et de la rétine sont déterminées génétiquement, alors que les connexions de l'aire V1 se développent en fonction des informations reçues du monde extérieur. Ceci est très probablement le cas aussi pour les autres aires du cortex visuel.



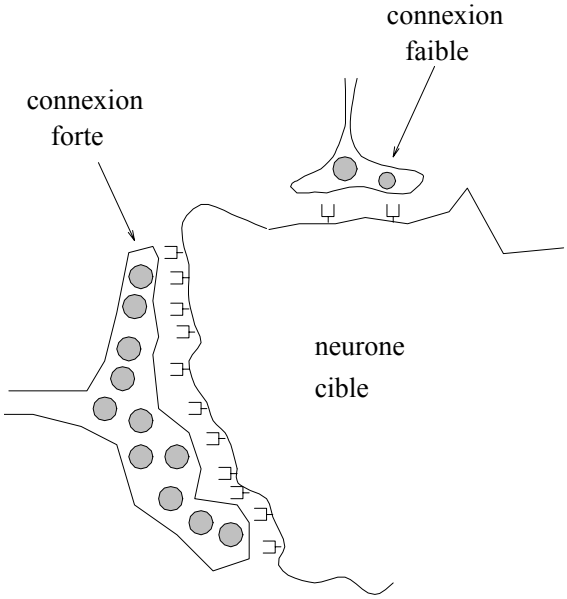
CONSOLIDATION D'UNE CONNEXION

Figure 15:

D'autres expériences, toujours chez le chat, confirment cette conclusion. L'un des yeux est privé de stimulations dès la naissance, puis rouvert au bout de 6 mois. On soumet alors cet œil à des stimulations visuelles et on mesure les réponses de divers neurones à ces stimulations. Dans la rétine et dans le corps genouillé, tous les neurones répondent correctement. Par contre, dans les aires visuelles du cortex, très peu de neurones répondent aux stimulations provenant de cet œil. Les neurones du cortex ont donc préférentiellement établi des connexions avec les fibres actives durant les premiers mois suivant la naissance.

3.4 L'ajustement de l'efficacité des connexions

Les modifications de l'efficacité des connexions synaptiques semblent être la principale source de l'apprentissage chez l'adulte. Le renforcement de l'efficacité d'une connexion peut être le résultat de plusieurs phénomènes (fig 16): l'augmentation de la surface de connexion, l'augmentation du nombre de vésicules de neuromédiateurs dans la terminaison synaptique, l'augmentation du nombre de récepteurs dendritiques. L'intervention d'une protéine kinase (CAM kinase II) a également été suggérée [*Alkon*89]. Sa migration du cytoplasme vers la membrane provoquerait des modifications de la perméabilité membranaire et rendrait le neurone cible plus facilement excitable.



FORCE DES CONNEXIONS SYNAPTIQUES

Figure 16:

La cause de ces modifications physiologiques est très mal connue. On a peu progressé depuis le modèle empirique de Hebb en 1949 [*Hebb*49]. Selon Hebb, l'efficacité d'une connexion se renforce lorsque le neurone émetteur et le neurone cible sont actifs ensemble et s'affaiblit lorsqu'ils sont désynchronisés. Mathématiquement, en se référant à notre modèle du neurone, ceci peut s'exprimer par:

$$\frac{dW_{ij}}{dt} = \alpha O_i O_j - \beta W_{ij}$$

Le premier terme du second membre traduit le fait que la connexion se renforce

d'autant plus rapidement que les neurones sont actifs ensemble. Le second terme traduit une décroissance naturelle de la force de la connexion en l'absence de signal ou en cas de désynchronisation. Il est certain que ce modèle est très incomplet, mais il a cependant l'intérêt de capturer certains principes de l'apprentissage.

4 Conclusion

Quels enseignements peut-on tirer des connaissances de la neurobiologie pour la conception de systèmes artificiels de vision ?

La première section nous a permis de dériver un modèle mathématique du neurone, qui a l'avantage d'être simple et aisément simulable sur ordinateur. Nous utiliserons extensivement ce modèle.

La section consacrée à l'organisation du système visuel humain met en évidence plusieurs points:

1. L'organisation hiérarchique.
2. La structure en couches, les premières couches étant génétiquement programmées.
3. Le parallélisme massif et les connexions locales.
4. L'évidence d'un traitement généralement non-itératif de l'information, avec au plus une dizaine d'étapes de calculs.

Ces éléments nous incitent à choisir de préférence un modèle de réseau de neurones non-bouclé, avec une structure en couches. De plus, comme on l'a souligné, un faible nombre de couches semble suffisant pour extraire des informations de haut niveau.

Nous réaliserons en général des prétraitements sur l'image afin de réduire le volume des données à traiter. Il est d'ailleurs intéressant de remarquer que les premières couches de neurones du système visuel humain réalisent effectivement une telle compression de données. De plus, les connexions des neurones de ces premières couches semblent être génétiquement programmées, et ne sont donc pas soumises à l'apprentissage.

Enfin, lorsque les données à analyser sont organisées selon une certaine topologie, on

exploitera cette topologie en établissant des connexions locales. On introduit ainsi une information a priori dans la structure même du réseau de neurones.

Pour ce qui est de l'apprentissage, nous avons malheureusement peu d'éléments, si ce n'est la certitude que la modification de l'efficacité des connexions nerveuses est en grande partie la source de l'apprentissage. On pourrait reprendre le modèle de Hebb, mais nous verrons qu'il est possible de dériver mathématiquement des règles d'ajustement des poids synaptiques qui semblent plus efficaces que la règle de Hebb. Ce sera l'objet du prochain chapitre.

Références

[*Aoki*88] Chiye AOKI, Philip SIEKEVITZ
"Plasticity in brain development"
Scientific American, December 1988

[*Alkon*89] Daniel L. ALKON
"Memory storage and neural systems"
Scientific American, July 1989

[*Crick*86] F. CRICK, C. ASANUMA
"Certain aspects of the anatomy and physiology of the cerebral cortex"
Parallel Distributed Processing, D.E. RUMELHART and J.L. Mc CLELLAND
Chap20, Bradford book - MIT Press - 1986

[*Daugman*89] John G. DAUGMAN
"Entropy reduction and decorrelation in visual coding
by oriented neural receptive fields"
IEEE Biomedical Engineering, vol 36, n^o 1, January 1989

[*Hebb*49] D.O. HEBB
"The organization of behavior"
J.WILEY and Sons, 1949

[*Hermann*79] H. HERMANN, J.F. CIER
"Physiologie générale du nerf"
Précis de physiologie (faculté de médecine),
Tome2, Chap8, Quatrième édition, MASSON, 1979

[*Imbert*83] Michel IMBERT
"La neurobiologie de l'image"
La Recherche, Mai 1983

[*Kennedy*87] Henri KENNEDY, Colette DEHAY
"Le développement du cerveau"
La Recherche, Juin 1990

[*Nathans*89] Jeremy NATHANS
"The genes for color vision"
Scientific American, February 1989

[*Piccolino*88] Marco PICCOLINO
"La vision et la dopamine"
La Recherche, Décembre 1988

[*Porat*89] Moshe PORAT, Yehoshua Y. ZEEVI
"Localized texture processing in vision:
analysis and synthesis in the Gaborian space"
IEEE Biomedical Engineering, vol 36, n^o 1, January 1989

[*Taxi*71] Jacques TAXI
"Comment fonctionne le système nerveux"
La Recherche, Janvier 1971

[*Thorpe*88] Simon J. THORPE
"Traitement d'images chez l'homme"
T.S.I. vol 7, n^o 6, 1988

[*Zeki*90] Semir ZEKI
"La construction des images par le cerveau"
La Recherche, Juin 1990

[*Watson*89] Andrew B. WATSON, Albert J. AHUMADA
"A hexagonal orthogonal-oriented pyramid as a model
of image representation in the visual cortex"
IEEE Biomedical Engineering, vol 36, n^o 1, January 1989

Chapitre 2:

LES MODELES NEURONAUX

Les réseaux de neurones formels sont des modèles théoriques de traitement de l'information inspirés des observations relatives au fonctionnement des neurones biologiques et du cortex cérébral.

Le domaine des réseaux de neurones n'est pas nouveau car il a son origine dans des travaux conduits durant les années 40 *(modèle de Hebb pour l'évolution des connexions synaptiques). Ces travaux conduisirent au modèle du perceptron dans les années 60 (modèle qui a principalement été appliqué à la reconnaissance de caractères). Mais ce n'est qu'à partir de 1986 que la recherche dans ce domaine a connu une expansion importante du fait de la publication de modèles de réseaux et d'algorithmes d'apprentissage suffisamment efficaces pour résoudre des problèmes réalistes et complexes.*

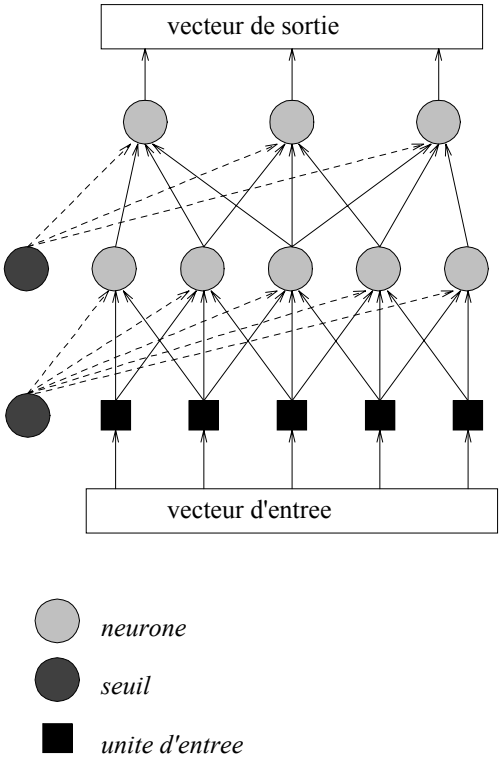
Depuis la parution de l'ouvrage de Rumelhart "Parallel Distributed Processing", dans lequel est notamment décrit l'algorithme de rétropropagation du gradient, un grand nombre de modèles a été proposé. Pourtant, la plupart d'entre eux se révèlent incapables de traiter des applications réelles. Ces algorithmes sont donc surtout intéressants pour éclairer certains aspects du fonctionnement cérébral, mais présentent peu d'intérêt pour le problème qui nous intéresse, à savoir la mise en œuvre d'applications de traitement du signal ou de l'image d'intérêt industriel ou opérationnel.

Compte tenu de cet objectif, et des points énoncés au chapitre précédent, nous avons porté notre attention sur 2 modèles: le perceptron multicouches (généralement associé à l'algorithme de rétropropagation) et le modèle de Kohonen (modèle des "cartes topologiques"). Ce chapitre est consacré à la description de ces modèles. Nous tenterons également de les situer par rapport à des théories mathématiques classiques: la théorie de la décision (ou plus exactement sa restriction à la classification) pour le perceptron multicouches, et la quantification vectorielle pour le modèle de Kohonen.

1 L'algorithme de rétropropagation

1.1 Le modèle et l'apprentissage

L'algorithme de rétropropagation a été proposé en 1986 par Rumelhart [*Rumelhart*86]. Il s'agit d'un algorithme d'apprentissage dédié aux réseaux multi-couches (fig 1) avec un modèle de neurone correspondant à la figure 2, sauf pour les neurones de la couche d'entrée qui n'effectuent aucun traitement.



PERCEPTRON MULTICOUCHES

Figure 1:

Le neurone est constitué d'un sommateur pondéré, suivi d'une non-linéarité. Ce modèle est donc globalement conforme aux observations biologiques. Les neurones sont organisés en couches:

- Une couche d'entrée qui reçoit les informations provenant de l'extérieur. Les neurones de cette couche correspondent par exemple aux neurones sensoriels

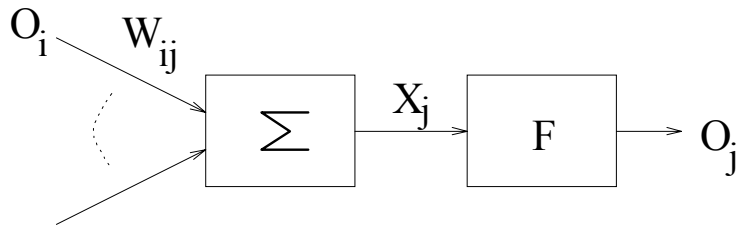


Figure 2: modèle du neurone

présents dans la rétine. Ils sont d'un type particulier car ils se contentent de transmettre l'information qui leur est présentée sans traitement.

- Une ou plusieurs couches intermédiaires, encore appelées couches cachées car elles ne sont pas directement en contact avec le monde extérieur.

- Une couche de sortie, correspondant aux neurones moteurs qui actionnent les muscles.

Il n'y a pas de connexions à l'intérieur d'une même couche. Chaque neurone reçoit ses entrées de la couche directement inférieure.

Posons :

O_j	= état du neurone j (valeur de sortie).
F	= fonction de transition du neurone.
W_{ij}	= coefficient de pondération associé à la connexion ij.
X_j	= potentiel du neurone (résultat de la somme pondérée).

Conformément à ces notations, nous avons:

$$X_j = \sum_i W_{ij} O_i$$

$$O_j = F(X_j)$$

Au départ, les poids W_{ij} sont initialisés aléatoirement. Lorsque l'on présente un vecteur d'entrée au réseau, il peut calculer par propagation couche après couche les états des neurones de la couche de sortie en utilisant les équations précédentes.
De même qu'une réponse musculaire doit être adaptée à la stimulation sensorielle reçue, il convient que les états des neurones de sortie soient adaptés à l'entrée présentée. On atteint cet objectif grâce à l'apprentissage. Pour cela, on se crée une

base d'exemples. Chaque exemple est constitué du vecteur d'entrée et du vecteur de sortie approprié. Ainsi, si l'on souhaite apprendre à un réseau de neurones à réaliser la fonction "OU EXCLUSIF", la base d'exemples sera:

exemple numéro	vecteur	d'entrée	vecteur de sortie
1	-1	-1	-1
2	-1	$+1$	$+1$
3	$+1$	-1	$+1$
4	$+1$	$+1$	-1

L'apprentissage consiste à adapter les poids des connexions (W_{ij}) afin d'obtenir le vecteur de sortie correct quel que soit l'exemple présenté. Pour cela, on se définit une fonction d'erreur (on choisit en général l'erreur quadratique moyenne). En notant O_j les sorties obtenues et S_j les sorties souhaitées, l'erreur quadratique et l'erreur quadratique moyenne sont:

$$e_Q = \frac{1}{2} \sum_{j \in sortie} (O_j - S_j)^2$$

$$e_{QM} = E\{e_Q\}$$

e_Q dépend de l'exemple présenté, alors que e_{QM} est une moyenne sur les exemples. Comme nous aurons besoin d'estimer des espérances mathématiques (ce qui sera réalisé grâce à des filtres passe-bas), les exemples seront présentés dans un ordre aléatoire.
On présente plusieurs fois tous les exemples dans un ordre aléatoire. A chaque présentation d'un exemple, on fait varier chaque poids W_{ij} dans le sens inverse du gradient de l'erreur :

$$\Delta W_{ij} = -\alpha \frac{\partial e_{QM}}{\partial W_{ij}}$$

α étant une constante positive, on s'assure ainsi une diminution de l'erreur.
Nous noterons à présent g_{ij} le gradient local de l'erreur quadratique:

$$g_{ij} = \frac{\partial e_Q}{\partial W_{ij}}$$

Comme $e_{QM} = E\{e_Q\}$, on peut écrire:

$$\Delta W_{ij} = -\alpha E\{\frac{\partial e_Q}{\partial W_{ij}}\}$$

car la dérivation est un opérateur linéaire. On a donc :

$$\Delta W_{ij} = -\alpha E\{g_{ij}\}$$

Une estimation $\bar{g}_{ij}$ de $E\{g_{ij}\}$ est obtenue par filtrage passe bas de g_{ij}. Utilisons un indice t qui est incrémenté chaque fois qu'un nouvel exemple est présenté:

$$\bar{g}_{ij}(t) = (1-\beta)g_{ij}(t) + \beta\bar{g}_{ij}(t-1)$$

Le terme d'oubli β est proche de 1 (typiquement 0.98). Rumelhart [*Rumelhart*86] a proposé une méthode rapide pour calculer les g_{ij}. Cette méthode consiste à propager un terme d'erreur δ_j de la sortie vers l'entrée du réseau (d'où le nom d'algorithme de rétropropagation).
De la définition de g_{ij} on déduit:

$$g_{ij} = \frac{\partial e_Q}{\partial X_j}\frac{\partial X_j}{\partial W_{ij}}$$

Posons:

$$\delta_j = \frac{\partial e_Q}{\partial X_j}$$

Alors:

$$\mathbf{g_{ij}} = \delta_\mathbf{j}\mathbf{O_i}$$

Et δ_j est déterminé comme suit:
$\Longrightarrow$ Si j est sur la couche de sortie:

$$\begin{aligned}\delta_j &= \frac{\partial e_Q}{\partial O_j}\frac{\partial O_j}{\partial X_j} \\ &= (O_j - S_j)F'(X_j)\end{aligned}$$

$\Longrightarrow$ Si j est sur une couche cachée:

$$\begin{aligned}\delta_j &= \frac{\partial e_Q}{\partial O_j}\frac{\partial O_j}{\partial X_j} \\ &= \left[\sum_{k\in succ(j)}\left(\frac{\partial e_Q}{\partial X_k}\frac{\partial X_k}{\partial O_j}\right)\right]\frac{\partial O_j}{\partial X_j} \\ &= \left[\sum_{k\in succ(j)} W_{jk}\delta_k\right]F'(X_j)\end{aligned}$$

où $succ(j)$ désigne l'ensemble des successeurs du neurone j.

1.2 Le choix de la fonction de transition

Le choix de la fonction de transition F est a priori laissé au concepteur de l'application neuronale. La seule contrainte impérative est que F soit dérivable. Toutefois, il est souhaitable de respecter les contraintes suivantes:

1. **La fonction est non-linéaire:**
   En effet, un réseau entièrement linéaire peut toujours se ramener à un réseau équivalent sans couche intermédiaire. Il suffit pour cela de faire le produit des matrices de transformation entre les différentes couches. On perd donc tout l'avantage d'une structure multicouches.

2. **La fonction est strictement croissante:**
   Cette condition s'exprime par $F'(X_j) > 0 \ \ \forall X_j \in \mathcal{R}$. Il est en effet souhaitable que la dérivée ne s'annule pas afin d'éviter un blocage de l'apprentissage (gradient nul). De plus, des changements de signe de la dérivée provoquent des inversions brutales du gradient qui déstabilisent l'apprentissage: la fonction doit donc être monotone. On peut sans perte de généralité imposer à F d'être croissante. En effet, si F est décroissante, il est toujours possible de la remplacer par la fonction croissante F_c telle que $F_c(X_j) = F(-X_j)$ et d'inverser les poids W_{ij}.

3. **La fonction est saturante:**
   Cette condition s'exprime par $\mid F(\pm\infty) \mid < \infty$. Elle correspond aux observations biologiques, et, de plus, la présence de saturations procure au réseau une forte robustesse par rapport au bruit. Enfin, la présence de saturations permet au neurone de fonctionner comme un calculateur binaire quand ceci est nécessaire (saturation haute=VRAI, saturation basse = FAUX).

4. **La fonction présente une zone quasi-linéaire:**
   Ceci peut être utile lorsque l'application requiert une certaine précision.

Dans le cadre de cette thèse, nous utiliserons toujours, sauf mention contraire, la fonction tangente hyperbolique:

$$\mathbf{F(X_j) = th(X_j)}$$

Cette fonction (fig 3) respecte les contraintes citées. De plus, elle présente des caractéristiques avantageuses:

- Présence d'une zone quasi-linéaire de part et d'autre de zéro. On peut considérer que dans cette zone le neurone fonctionne comme un dispositif analogique.

- Présence de saturations très proches de la zone quasi-linéaire. On peut considérer que dans cette zone de saturation, le neurone fonctionne comme un dispositif numérique (binaire).

Ainsi, en raisonnant qualitativement, on peut dire que le neurone a la possibilité d'ajuster ses poids pour fonctionner en analogique ou en numérique suivant les besoins.

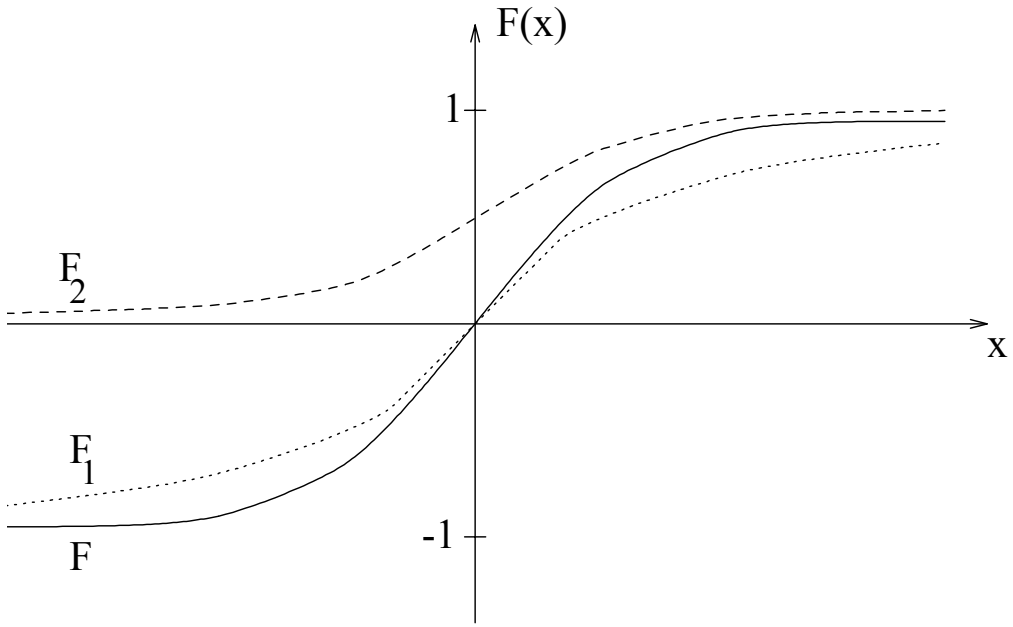


Figure 3: fonctions de transition

Nous avons mené (sur des problèmes de classification) des expériences comparatives avec d'autres fonctions. Les résultats obtenus avec la tangente hyperbolique ont toujours été nettement supérieurs: apprentissage plus rapide (souvent d'un ou plusieurs ordres de grandeur) et meilleure généralisation. Par exemple, la fonction:

$$F_1(X_j) = \frac{X_j}{1 + |X_j|}$$

dont l'allure (fig 3) est pourtant très proche de $th(X_j)$ conduit souvent à des résultats médiocres. Ceci est probablement dû au fait que la saturation est trop éloignée de la zone quasi-linéaire, d'où la difficulté pour le neurone de changer de mode de fonctionnement.

La fonction (fig 3):

$$F_2(X_j) = \frac{1}{2}\left(1 + th(X_j)\right)$$

conduit à un apprentissage plus long que $th(X_j)$. Il semble donc que la symétrie par rapport à l'origine soit un élément accélérateur de l'apprentissage. Nous n'avons pas

recherché en détail les causes théoriques de ce phénomène. Toutefois, en considérant le vecteur dont les composantes sont les états des neurones d'une couche, on peut faire une remarque intéressante. Lorsque tous les neurones saturent, la norme de ce vecteur est constante si la fonction de transition du neurone est F car toutes ses composantes sont ± 1. Par contre cette norme est variable dans le cas de F_2 car les composantes sont 0 ou 1, et la norme dépend du nombre de 1. Ceci est probablement un élément de déstabilisation de l'apprentissage car les neurones de la couche supérieure reçoivent des vecteurs d'entrée dont la norme varie avec l'exemple présenté.

La dérivée de la fonction F peut être aisément calculée en fonction de la sortie du neurone (en utilisant les propriétés de la tangente hyperbolique):

$$\mathbf{F'(X_j) = 1 - O_j^2}$$

2 Relations avec la théorie de la classification

2.1 Généralités

Le problème de la classification consiste à décider à quel sous-ensemble d'une partition appartient une observation donnée [*Arquès*82]. La problématique générale est représentée sur la figure 4. On dispose d'une observation X dont la loi de probabilité dépend d'un paramètre discret θ (classe de X). Le problème est d'associer à X une décision δ concernant sa classe. Ces variables appartiennent à 3 ensembles: Θ, χ *et* Δ. Δ est en correspondance bijective avec Θ, et l'objectif est bien entendu de faire en sorte que la décision δ soit aussi souvent que possible celle qui correspond à la vraie classe.

La prise de décision se fait via une fonction de décision ϕ appartenant à l'ensemble Φ des fonctions possibles. Le problème revient à déterminer la meilleure fonction de décision, au sens d'un certain critère.

Tout classifieur possède normalement un certain nombre de paramètres internes: les poids des connexions dans le cas des classifieurs neuronaux ou des paramètres relatifs aux distributions de probabilité dans le cas des classifieurs Bayésiens. Ces paramètres doivent être ajustés en fonction des données expérimentales. C'est pourquoi, on

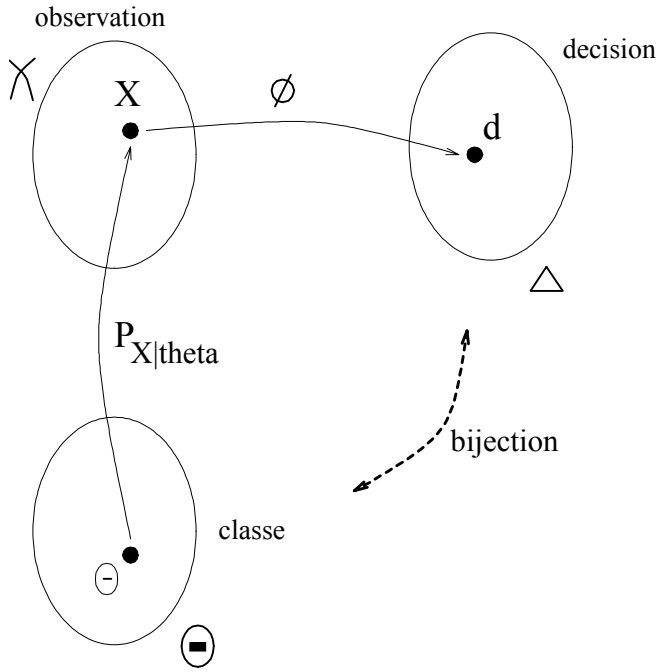


Figure 4: Modélisation d'un problème de classification

distingue généralement deux étapes dans une application de classification:

1. L'ajustement des paramètres du classifieur sur un ensemble d'apprentissage.
2. Le test de ses performances sur un ensemble d'évaluation.

L'ensemble d'apprentissage contient des observations ainsi que la connaissance de leur vraie classe. On ajuste alors les paramètres du classifieur en fonction de ces exemples. Puis on peut mesurer le pourcentage de classifications correctes sur un ensemble d'évaluation. Ce dernier ne doit pas contenir d'exemples ayant servi à l'apprentissage sans quoi les mesures seraient faussées (il y a en effet peu de chances pour qu'en situation réelle le classifieur retrouve exactement des observations identiques à celles de l'apprentissage).

La recherche dans le domaine de la classification est importante depuis les années 60, et a en grande partie portée sur les classifieurs Bayésiens. Ce sont toujours les classifieurs les plus largement utilisés, probablement du fait qu'ils sont extensivement décrits dans nombre d'ouvrages. Ils ont de plus l'avantage de reposer sur des bases mathématiques solides, et sont optimaux lorsque l'on connait la distribution de probabilité de l'observation conditionellement à la classe, ainsi que les probabilités a priori des diverses classes.

Nous décrivons brièvement ce type de classifieur dans le paragraphe suivant afin de fixer le cadre mathématique. Mais nous verrons que les connaissances requises sur les distributions de probabilité sont très rarement disponibles. Dans le cas général, la distribution de probabilité de l'observation conditionnellement à la classe a une forme extrêmement complexe. Les utilisateurs de classifieurs Bayésiens contournent ce problème en extrayant de l'observation des caractéristiques qui sont supposées avoir une loi de probabilité conditionnelle simple et en faisant des hypothèses plus ou moins arbitraires sur la forme exacte de cette loi (gaussienne,...). En conséquence, le classifieur Bayésien n'est souvent pas optimal car la distribution assumée ne correspond pas à la distribution réelle.

Les remarques précédentes permettent de comprendre pourquoi d'autres types de classifieurs non probabilistes, et en particulier les classifieurs neuronaux sont souvent plus performants. Après avoir présenté les classifieurs Bayésiens, nous situerons les réseaux de neurones multicouches dans le cadre de la classification.

2.2 Classifieurs Bayésiens

Les classifieurs Bayésiens supposent connue la distribution de probabilité de l'observation conditionnellement à la classe ($P_{X|\theta}$), ainsi que les probabilités a priori (P_θ) des diverses classes. Cette connaissance permet de choisir la fonction de décision ϕ optimale au sens d'un certain critère. L'élaboration d'un critère nécessite la définition d'une fonction de coût $C(\theta, \delta)$: coût du choix de la classe δ lorsque la vraie classe est θ. Une fonction de coût souvent utilisée est :

$$\begin{aligned} C(\theta,\delta) &= 0 \quad si\ \theta = \delta \\ &= 1 \quad sinon \end{aligned}$$

On définit alors :

- Le risque a priori, qui correspond à l'espérance du coût pour X et ϕ fixés :

$$\rho(\phi, X) = E_{(\theta,\delta)|X}\{C(\theta,\delta)\}$$

- Le risque a posteriori, qui correspond à l'espérance du coût pour θ et ϕ fixés :

$$r(\phi, \theta) = E_{(X,\delta)|\theta}\{C(\theta,\delta)\}$$

- Le risque moyen, qui correspond à l'espérance du coût pour ϕ fixé:

$$R(\phi) = E_{X,\theta,\delta}\{C(\theta,\delta)\}$$

On peut montrer que $R = E_X\{\rho\} = E_\theta\{r\}$. Cherchons par exemple la fonction ϕ optimale au sens du risque moyen (c'est à dire celle qui minimise le risque moyen). On peut écrire:

$$\begin{aligned} R(\phi) &= \textstyle\sum_\theta E_{(X,\delta)|\theta}\{C(\theta,\delta)\}P_\theta(\theta) \\ &= \textstyle\sum_\theta \left\{\sum_X E_{\delta|(X,\theta)}\{C(\theta,\delta)\}P_{X|\theta}(X \mid \theta)\right\} P_\theta(\theta) \end{aligned}$$

Si ϕ est déterministe, l'action de ϕ revient à faire une partition de χ en domaines D_δ (ensemble des X qui ont pour image δ). On peut alors écrire :

$$\begin{aligned} R(\phi) &= \textstyle\sum_\theta \left[\sum_\delta C(\theta,\delta)\left[\sum_{X\in D_\delta} P_{X|\theta}(X \mid \theta)\right]\right] P_\theta(\theta) \\ &= \textstyle\sum_\delta \sum_{X\in D_\delta} \sum_\theta C(\theta,\delta)P_{X,\theta}(X,\theta) \end{aligned}$$

Par conséquent, X doit être affecté au domaine D_δ tel que $\sum_\theta C(\theta,\delta)P_{X,\theta}(X,\theta)$ soit minimal.

Le problème est qu'il faut connaître les lois $P_{X|\theta}$ et P_θ. Il est généralement possible d'estimer P_θ, à partir des données expérimentales. Par exemple, pour un problème de reconnaissance de caractères manuscrits, on peut mesurer la fréquence d'apparition des différents caractères. Par contre, la connaissance de $P_{X|\theta}$ n'est généralement pas possible, sauf dans des cas très simples. Par exemple, lorsque l'observation est une fenêtre extraite d'une image (comme c'est le cas pour la classification de texture), ceci reviendrait à connaître la probabilité de chaque réalisation possible dans la fenêtre d'observation! Or cette connaissance est primordiale car c'est cette loi qui caractérise le phénomène observé. On tente généralement de surmonter ce problème en calculant à partir de l'observation X un vecteur Y de faible dimension (extraction de caractéristiques). Les composantes de Y sont choisies pour être fortement discriminantes. Par exemple, pour la classification de texture, une des composantes de Y pourrait être la variance de la luminance dans la fenêtre d'observation. Si Y est bien choisi, on peut supposer que $P_{Y|\theta}$ a une forme gaussienne. Les coefficients caractérisant la gaussienne peuvent alors être estimés à partir des données expérimentales.

L'inconvénient de cette approche est d'une part le fait qu'une analyse poussée de l'application est nécessaire afin de déterminer les bonnes caractéristiques à extraire de l'observation (elles doivent en particulier respecter une loi de probabilité simple, de préférence gaussienne), et d'autre part le fait que l'on risque de perdre de l'information pertinente dans le processus d'extraction de caractéristiques.

2.3 Classifieurs non probabilistes

2.3.1 Définition

Nous avons vus au paragraphe précédent que le classifieur Bayésien effectue une partition de l'ensemble χ en domaines de décision D_δ. Cette partition est optimale lorsque l'on connait la distribution de l'observation conditionnellement à la classe.

Pour résumer, la mise au point d'un classifieur Bayésien est la suivante:

1. Estimation des probabilités a priori P_θ.
2. Choix d'une forme paramétrique de la loi conditionnelle $P_{X|\theta}$.
3. Estimation des paramètres de cette loi conditionnelle à partir des données expérimentales.
4. Utilisation de ces lois pour déterminer les domaines de décision D_δ

Le point faible de cette démarche est le choix de la forme paramétrique de $P_{X|\theta}$, car il ne peut souvent être fait que de façon plus ou moins arbitraire. Or, un mauvais choix peut conduire à des performances désastreuses. D'où l'idée de classifieurs qui détermineraient directement des domaines de décision sans passer par l'intermédiaire des lois de probabilité. Lorsqu'ils sont utilisés à des fins de classification, les perceptrons multicouches se situent dans cette catégorie.

Les classifieurs non probabilistes possèdent des paramètres internes qui leur permettent d'ajuster les régions de décision aux données expérimentales. L'espace χ est supposé être un espace métrique, ce qui est le cas pour la plupart des applications. Ils peuvent être distingués suivant la forme des régions de décision qu'ils sont capables de créer. Nous ne passerons pas ici en revue les différents classifieurs existants (une telle revue pourra être trouvée dans [*Lippmann*89]), mais nous décrivons brièvement ci-dessous trois catégories importantes:

2.3.2 Les classifieurs par hyperplans

Les classifieurs par hyperplans découpent l'ensemble χ à l'aide d'hyperplans. Les perceptrons multicouches peuvent être situés dans cette catégorie. Les poids des

connexions synaptiques permettent d'ajuster l'orientation et la position des divers hyperplans. Nous verrons que sous réserve d'un nombre de couche au moins égal à 4, un perceptron multicouches est capable de réaliser une partition arbitrairement complexe de l'ensemble χ.
Les classifieurs par arbre de décision (fig 5) se situent également dans cette catégorie. Les hyperplans sont alors contraints à rester parallèles aux axes, ce qui n'est pas le cas pour un réseau de neurones. Un exemple de classification par arbre de décision est fourni figure 5 (pour un cas à 2 classes [carrés blancs/ronds noirs], et un espace paramétrique de dimension 2).

Les classifieurs par hyperplans nécessitent souvent un apprentissage assez long pour ajuster leurs paramètres internes, mais sont ensuite très rapides en reconnaissance, et peu gourmands en capacité mémoire. On décrit en annexe 1 deux classifieurs par hyperplans (la méthode de Hebb et la pseudo-inverse), que nous utiliserons souvent comme base de comparaison dans les chapitres applicatifs.

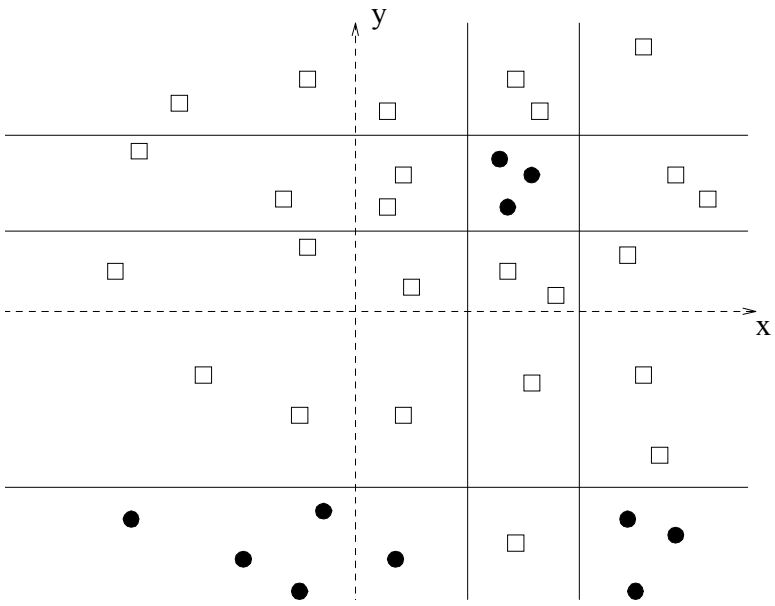


Figure 5: classification par arbre de décision

2.3.3 Les classifieurs par hypersphères

Les régions de décision créées par ce type de classifieur ont la forme de boules (fig 6). Chaque région est donc caractérisée par son centre et son rayon. Ce type de

classifieur est relativement rapide en apprentissage et en reconnaissance, mais réagit très mal lorsque les données sont bruitées. Il crée alors un grand nombre de boules et perd en rapidité et en performances.

Comme il n'est pas possible de réaliser un pavage complet et disjoint de l'espace à l'aide de boules, ces classifieurs présentent des régions de "non-décision" (points n'appartenant à aucune boule) et des régions ambigües (intersection de boules).

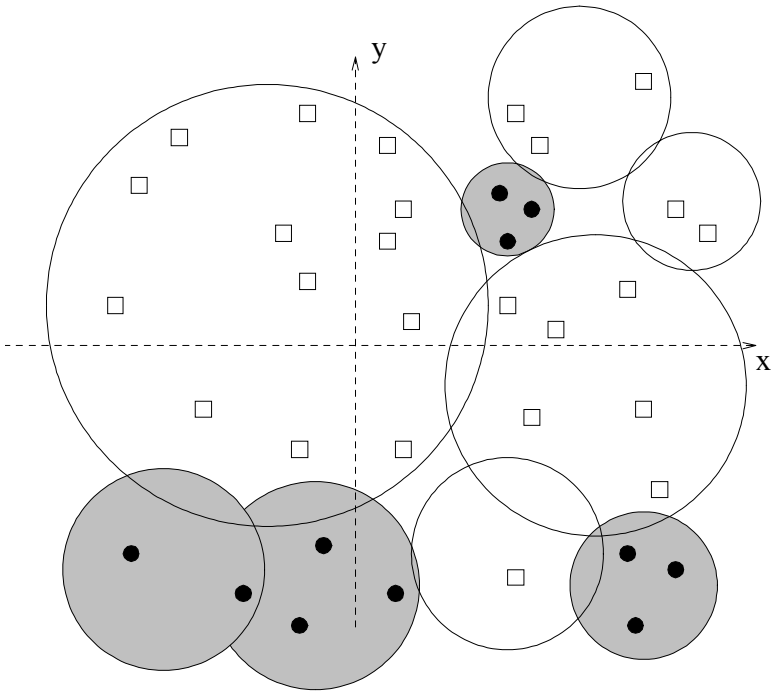


Figure 6: classification par hypersphères

2.3.4 Les classifieurs par nœuds

Ce type de classifieur utilise des nœuds pour caractériser les régions et affecte à tout point X la classe majoritaire parmi les nœuds les plus proches. Les nœuds sont généralement les exemples de l'ensemble d'apprentissage. Par exemple, le classifieur selon les k-plus proches voisins affecte tout point X de χ au domaine D_δ tel que δ est la classe majoritaire pour les k exemples de l'ensemble d'apprentissage les plus proches de X (au sens de la norme euclidienne).

L'apprentissage avec ce type de classifieur est très rapide (il suffit de stocker les exemples pour le classifieur selon les k-plus proches voisins). Mais la reconnaissance est généralement très coûteuse en temps de calcul du fait de la nécessité de balayer

un grand nombre de nœuds afin de déterminer le ou les plus proches voisins.

Contrairement à une idée reçue, les performances des classifieurs par nœuds sont généralement très bonnes, mais leur coût en puissance de calcul et en capacité mémoire est souvent rédhibitoire. Nous verrons que les classifieurs neuronaux atteignent des performances équivalentes et même souvent supérieures pour une puissance de calcul requise bien plus faible.

2.4 Le perceptron multicouches comme classifieur

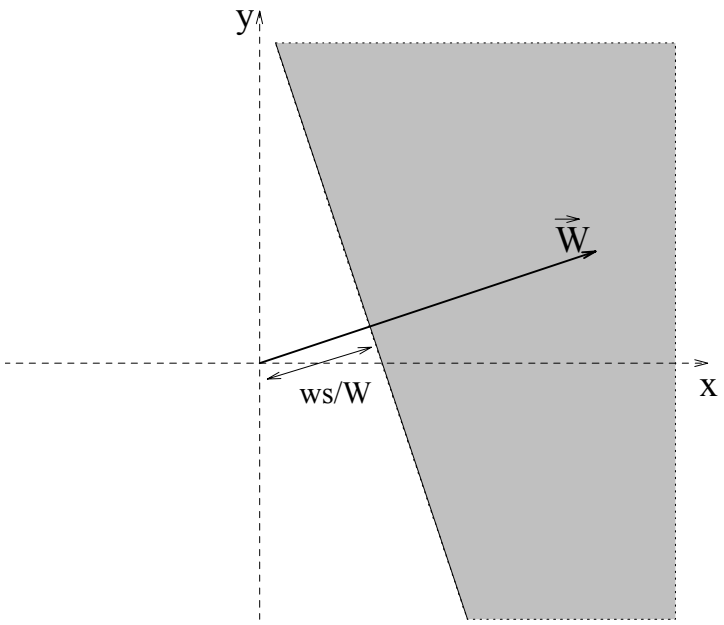


Figure 7: Domaines de décision d'un neurone binaire

Nous avons vu au §1 la structure du perceptron multi-couches. Pour une application de classification, la stratégie la plus simple consiste à disposer en sortie autant de neurones qu'il y a de classes. Chaque neurone correspond à une classe. Il doit répondre $+1$ lorsque le vecteur présenté en entrée appartient à sa classe, et -1 dans le cas contraire. En fait ces valeurs ne seront jamais atteintes exactement car elles correspondent aux valeurs assymptotiques de la tangente hyperbolique. On pourra donc se fixer une tolérance, par exemple 0.2. On arrêtera alors l'apprentissage lorsque (pour tous les exemples) toutes les sorties pour lesquelles on demande $+1$ sont supérieures à 0.8 et toutes les sorties pour lesquelles on demande -1 sont inférieures à -0.8, ou bien lorsque l'erreur quadratique moyenne ne diminue plus

suffisamment. Après apprentissage, pour classifier une observation X, il suffit de la présenter en entrée du réseau. On calcule les sorties du réseau et on décide de la classe correspondant au numéro du neurone qui produit la plus forte sortie. Par exemple, pour un problème à 4 classes, si l'on obtient en sortie (-0.9, 0.15, 0.7, 0.35), on décidera que la classe est 3 car c'est le troisième neurone qui présente la plus forte sortie.

Il est également possible de définir une confiance comme étant la différence entre la plus forte sortie, et la sortie immédiatement inférieure. On divisera cette valeur par 2 afin de se ramener à une confiance toujours comprise entre 0 et 1. Dans l'exemple précédent, la confiance est $(0.7\text{-}0.35)/2 = 0.175$.

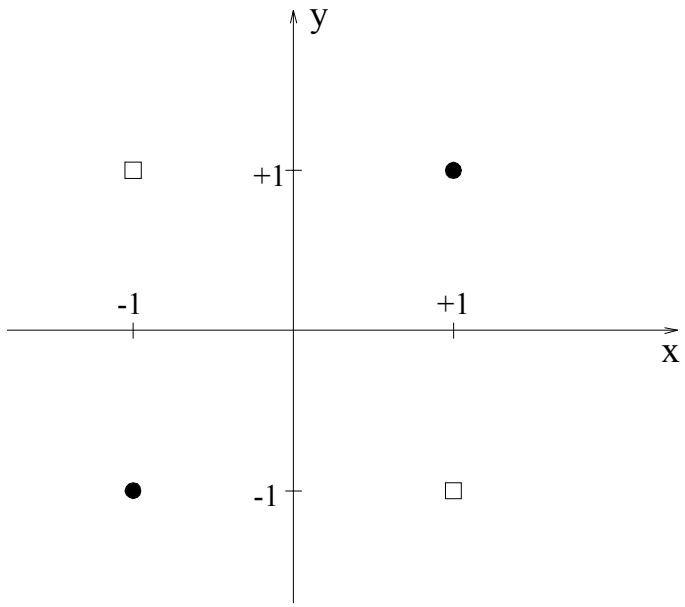


Figure 8: Exemple de classification non-réalisable avec un seul hyperplan

Il est intéressant d'étudier le type de régions de décision que peut former un perceptron multicouches. Pour cela, on s'intéresse d'abord à un neurone isolé, et on fait l'approximation d'une fonction de transition binaire: la fonction signe $sgn(X)$. Notons que le neurone dont la fonction de transition est la tangente hyperbolique peut s'approcher a volonté de ce type de comportement simplement en augmentant ses poids. Nous illustrons notre raisonnement (fig 7) dans le cas où l'on a seulement 2 connexions incidentes sur ce neurone, et une connexion provenant du neurone seuil; mais on peut aisément généraliser. Lorsque l'on présente un vecteur $\vec{e}$ en entrée d'un neurone dont les poids sont $\vec{W}$, son potentiel est $X = \vec{W}.\vec{e} + w_s$, où w_s est le poids de la connexion le liant au neurone seuil. Donc, si la projection de l'entrée sur $\vec{W}$ est

supérieure à $-\frac{w_s}{\|\vec{W}\|}$ alors $sgn(X)$ vaut $+1$, et -1 dans le cas contraire. Les régions de décision du neurone sont donc des demi-plans (ou des demi-espaces dans le cas général).

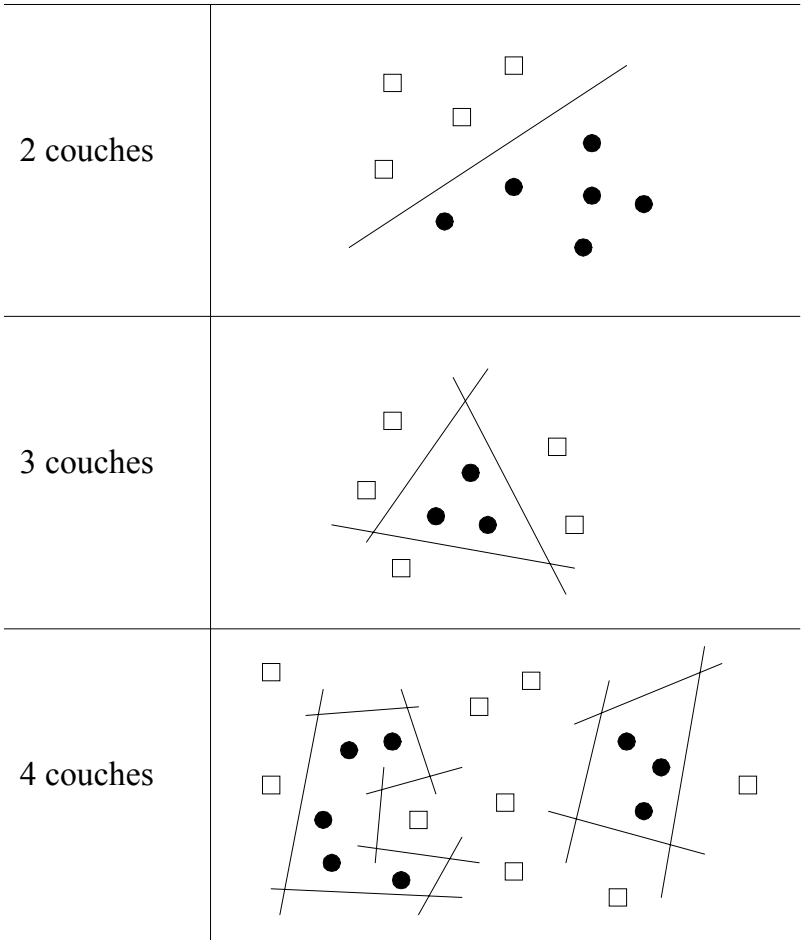


Figure 9: Domaines de décision d'un perceptron multicouches

Lorsque l'on a trois couches de neurones, la première couche cachée peut créer autant de droites séparatrices qu'elle a de neurones (n_1). Sa sortie est un code binaire de longueur n_1, c'est à dire l'un des sommets de l'hypercube de dimension n_1. Un neurone de la troisième couche pourra donc séparer les points de cet hypercube suivant un hyperplan. Tous les découpages possibles ne sont pas réalisables à ce niveau: par exemple, sur l'hypercube de dimension 2 il n'est pas possible de regrouper les sommets selon la figure 8 à l'aide d'un hyperplan. On peut montrer que les zones de décision créées au niveau de la troisième couche ne peuvent être que

convexes [*Lipmann*88]. Par contre, au niveau de la quatrième couche il est possible de réaliser un découpage arbitraire (fig 9).

3 Le modèle de Kohonen

Le modèle de Kohonen trouve son inspiration dans une structure locale particulière existant dans certaines aires du cortex (fig 10). Les neurones sont organisés en couches, et à l'intérieur de chaque couche, chaque neurone émet vers ses voisins les plus proches des connexions excitatrices et vers les neurones plus éloignés des connexions inhibitrices. Tous les neurones reçoivent les mêmes entrées.

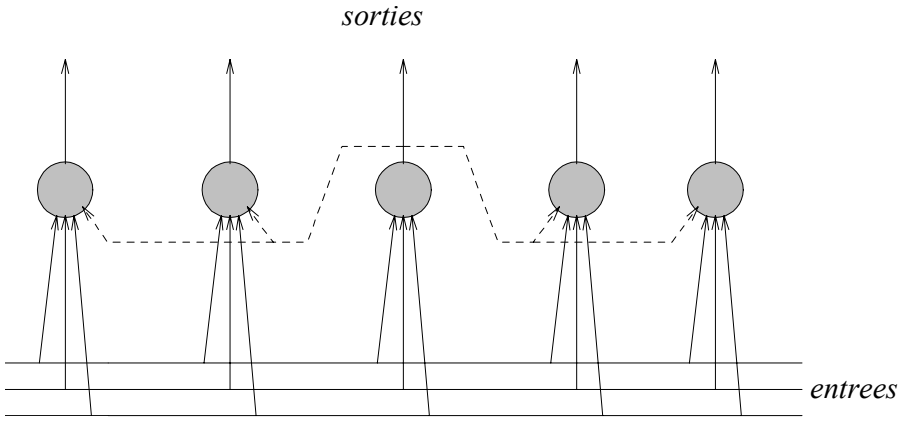


Figure 10: Modèle de Kohonen (1D)

Kohonen a simulé le fonctionnement de ce type de réseau [*Kohonen*84], en utilisant un modèle de neurone qui répond d'autant plus fortement que le vecteur d'entrée est proche de son vecteur poids. On peut prendre par exemple le modèle suivant (où O_j est la sortie du neurone, $\vec{W}_j$ son vecteur poids, et $\vec{x}$ le vecteur présenté à l'entrée):

$$O_j = \frac{1}{1 + \|\vec{W}_j - \vec{x}\|^2}$$

Lorsque l'on présente une entrée, les sorties évoluent sous l'effet des connexions intra-couche, puis se stabilisent rapidement de telle sorte que le neurone qui répondait

initialement le mieux à l'entrée, ainsi que ses voisins soient actifs, alors que les autres neurones sont inhibés.
Kohonen en a déduit un algorithme d'apprentissage qui permet de bien approximer ce comportement sans perdre de temps à calculer les diverses interactions latérales. Considérons un réseau de M neurones, dans lequel tous les neurones reçoivent le même vecteur d'entrée $\vec{x} = [x_1, x_2, ..., x_K]^T$. Le neurone qui répondra le plus fortement (neurone vainqueur) sera celui pour lequel $\|\vec{W}_j - \vec{x}\|^2$ est minimal. On peut simplifier le modèle du neurone car il suffit d'être capable de déterminer le vainqueur. La sortie d'un neurone j est alors:

$$O_j = \sum_{i=1}^{K} (W_{ij} - x_i)^2 = \|\vec{W}_j - \vec{x}\|^2$$

L'algorithme d'apprentissage est le suivant (on désigne par t l'indice d'itération):

1. Initialiser les poids aléatoirement
2. Présenter un vecteur d'entrée $\vec{x}$
3. Calculer les valeurs de sortie O_j pour tous les neurones
4. Déterminer le neurone j_0 dont la valeur de sortie O_{j_0} est la plus faible
5. Modifier les poids du vainqueur j_0 et de ses voisins conformément à:

$$W_{ij}(t+1) = W_{ij}(t) + \alpha(t, |j - j_0|).[x_i(t) - W_{ij}(t)]$$

6. Si nombre d'itérations accordé n'est pas atteint, aller en (2)

Le terme $\alpha(t, |j - j_0|)$ est un réel positif qui détermine la vitesse de variation des poids. Kohonen suggère de prendre :

$$\begin{aligned} \alpha(t, |j - j_0|) &= \alpha_0 \quad si\ |j - j_0| < V(t) \\ &= 0 \quad sinon \end{aligned}$$

Le terme α_0 est nommé "vitesse d'apprentissage". Kohonen suggère des valeurs de l'ordre de 10^{-1}. Le terme $V(t)$ est l'étendue du voisinage. Cette étendue est réduite en cours d'apprentissage pour accélérer l'organisation du réseau.

Nous proposerons au chapitre 4 une autre expression de α qui conduit généralement à de meilleures performances.

Il est clair qu'à chaque itération, le vainqueur j_0 et ses voisins vont déplacer leur vecteur poids en direction de l'entrée $\vec{x}$. Par conséquent, la dynamique du réseau est en quelque sorte le résultat d'une force extérieure (adaptation aux vecteurs d'entrée) et d'une force intérieure (les relations de voisinage, qui obligent des neurones proches à avoir des poids voisins). Kohonen a validé son algorithme sur des applications de traitement de la parole, et a montré que, lorsque le réseau est soumis à des échantillons de parole, les neurones adaptent leurs poids pour devenir représentatifs des phonèmes. De plus, les relations topologiques entre phonèmes sont conservées car des neurones proches sont sensibles à des phonèmes de consonance voisine.

On peut illustrer les propriétés du réseau de Kohonen sur un exemple simple. Supposons que le vecteur d'entrée soit de dimension 2 et que ses composantes correspondent aux coordonnées d'un point tiré au hasard à l'intérieur d'un triangle On réalise l'apprentissage en présentant un grand nombre de tels vecteurs. Comme chaque neurone a deux poids, on peut représenter les neurones par des points du plan. A l'issue de l'apprentissage, on constate qu'ils respectent une disposition du type de celle qui est représentée sur la figure 11. Le réseau de neurones, qui peut être considéré comme une courbe monodimensionnelle a donc réalisé une approximation d'une partie d'un espace à deux dimensions (le triangle).

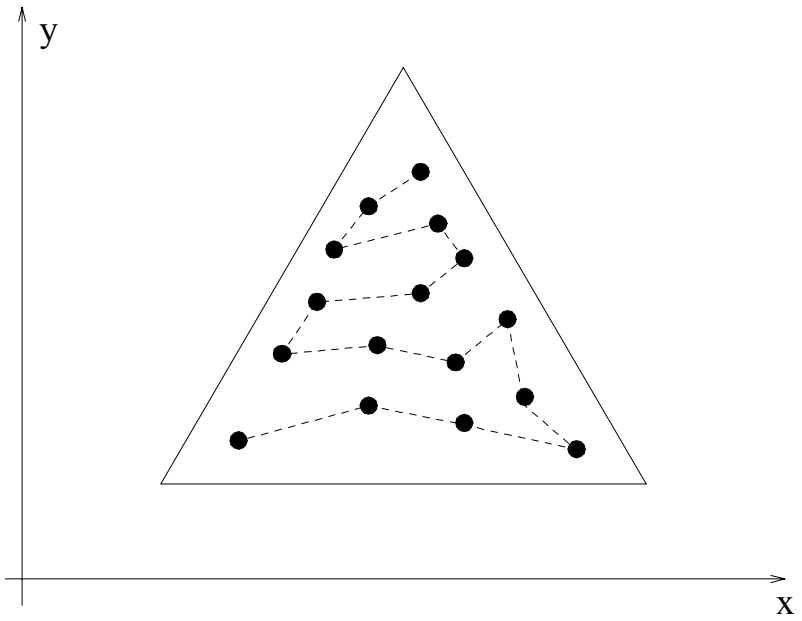


Figure 11: Auto-organisation 1D dans un triangle

L'algorithme de Kohonen peut également s'appliquer pour un réseau de neurones de dimension supérieure. Par exemple, pour un réseau à deux dimensions, les neurones sont distribués dans un plan et les relations de voisinage correspondent au voisinage bidimensionnel (fig 12).

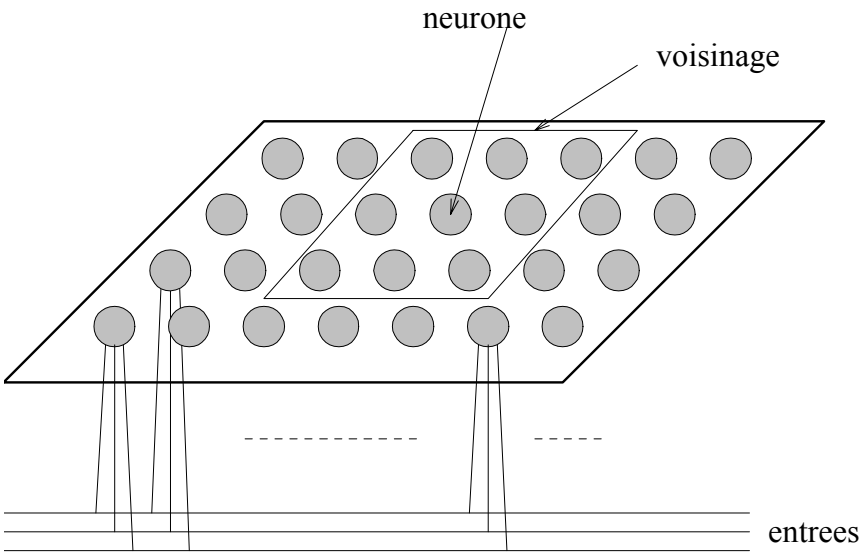


Figure 12: Modèle de Kohonen à 2 dimensions

4 La quantification vectorielle

4.1 Définition

Nous avons étudié précédemment (§2) la classification supervisée. Dans le cas supervisé, on connait pendant l'apprentissage la classe de chaque exemple. On crée donc des régions de décision conformément à cette information.

La classification non supervisée (ou quantification vectorielle) consiste à regrouper les exemples sans avoir aucune indication de classe. La seule possibilité est donc de les regrouper en classant ensemble les exemples qui sont proches au sens d'une certaine métrique (souvent la métrique Euclidienne).

Une revue de la quantification vectorielle est proposée dans [*Gray*84]. Conformément aux remarques précédentes, l'ensemble χ doit être muni d'une métrique. Généralement, χ est l'espace vectoriel $\Re^K$ muni de la métrique euclidienne. Le nombre de classes doit être connu a priori. Signalons que des méthodes permettant de déterminer le nombre de classes ont été proposées, mais nous n'en discuterons pas ici car ce problème ne se pose pas pour nos applications.

Nous noterons Δ le nombre de classes, et K la dimension de l'espace χ. Un algorithme de quantification vectorielle conduit à découper l'espace χ en domaines de décision D_δ. A chaque domaine D_δ est associé un "prototype" $\vec{W}_\delta$ qui doit être représentatif de ce domaine. On peut donc définir une fonction q qui à tout vecteur $\vec{x}$ associe un prototype:

$$q(\vec{x}) = \vec{W}_{classe(\vec{x})}$$

Cette fonction q peut être décomposée en un encodeur (classifieur) et un décodeur (recherche du prototype de la classe):

$$\begin{array}{rccccc} q : \Re^K & \longrightarrow & \{1, 2, ..., \Delta\} & \longrightarrow & \Re^K \\ \vec{x} & \longrightarrow & \delta & \longrightarrow & \hat{\vec{x}} = q(\vec{x}) = \vec{W}_\delta \end{array}$$

Pour une application de compression d'image, ce vecteur $\vec{x}$ sera un bloc d'image (par exemple un bloc 3x3). Donc, l'encodage transforme le vecteur d'entrée $\vec{x}$ en un nombre δ, et le décodage utilise δ pour générer le vecteur $\hat{\vec{x}} = \vec{W}_\delta$.
Pour la compression, une image est décomposée en blocs carrés disjoints, et chaque bloc est encodé. Pour la reconstruction, chaque bloc est remplacé par le prototype correspondant.

Notre objectif est bien entendu que $\hat{\vec{x}}$ soit une bonne représentation de $\vec{x}$, au sens de la métrique choisie. Si l'on utilise la métrique euclidienne, une bonne mesure de la qualité de la quantification réalisée est l'erreur quadratique moyenne:

$$e_{QM} = E\left\{\|\vec{x} - \hat{\vec{x}}\|^2\right\}$$

L'espérance mathématique sera en pratique remplacée par une moyenne sur un ensemble d'apprentissage $\mathcal{A}$. Cet ensemble est supposée assez grand pour bien représenter les statistiques du problème.

$$e_{QM} \simeq \frac{1}{card(\mathcal{A})} \sum_{\vec{x} \in \mathcal{A}} \|\vec{x} - \hat{\vec{x}}\|^2$$

On ne connait pas d'algorithme conduisant toujours à la solution optimale. La seule stratégie connue consiste à résoudre alternativement les 2 problèmes suivants:

1. Quand les prototypes $\vec{W}_\delta$ sont fixés, comment choisir au mieux les domaines D_δ?

2. Quand les domaines D_δ sont fixés, comment positionner au mieux les prototypes $\vec{W}_\delta$?

Solution au problème 1:

Pour minimiser l'erreur quadratique, le vecteur $\hat{\vec{x}}$ doit être le prototype le plus proche de $\vec{x}$. Les domaines D_δ doivent donc être tels que:

$$D_\delta = \left\{ \vec{x} \in \mathcal{A} \; tels \; que \; \forall \gamma \neq \delta, \|\vec{x} - \vec{W}_\delta\|^2 < \|\vec{x} - \vec{W}_\gamma\|^2 \right\}$$

Solution au problème 2:

L'erreur quadratique moyenne peut s'écrire:

$$e_{QM} = \frac{1}{card(\mathcal{A})} \sum_{\delta=1}^{\Delta} \sum_{\vec{x} \in D_\delta} \|\vec{x} - \vec{W}_\delta\|^2$$

Le prototype $\vec{W}_\delta$ doit donc être le vecteur qui minimise $\sum_{\vec{x} \in D_\delta} \|\vec{x} - \vec{W}_\delta\|^2$. On peut aisément montrer que la solution est le barycentre de la classe δ:

$$\vec{W}_\delta = \frac{1}{card(D_\delta)} \sum_{\vec{x} \in D_\delta} \vec{x}$$

Algorithme:

Ceci nous conduit à l'algorithme d'apprentissage suivant, connu sous le nom d'algorithme des "k-means" ([$MacQueen67$] cité dans [$Linde80$]). On note t l'indice d'itération.

1. Initialisation: choix aléatoire (ou guidé) de $\{\vec{W}_1, \vec{W}_2, ..., \vec{W}_\Delta\}$

2. Mettre à jour les classes D_δ

3. Calculer $e_{QM}(t)$ et arrêter si $\frac{e_{QM}(t) - e_{QM}(t-1)}{e_{QM}(t-1)} < \epsilon$, où ϵ est un réel positif.

4. Remplacer chaque prototype $\vec{W}_\delta$ par le centre de gravité de D_δ.

5. Aller en (2)

On constate aisément que l'erreur quadratique moyenne ne peut que diminuer à chaque itération. En effet, l'étape 2 produit forcément une réduction de l'erreur quadratique moyenne car elle consiste à mettre à jour le découpage de χ conformément à la solution optimale lorsque les prototypes sont fixés. Il en est de même pour l'étape 4 car lorsque les domaines D_δ sont fixés, le meilleur choix possible pour un prototype $\vec{W}_\delta$ est de le prendre au barycentre de sa classe.

Par contre, on n'est pas à l'abri d'un piégeage dans un minimum local de l'erreur quadratique. Pour réduire ce risque, Linde, Buzo et Gray ont proposé un algorithme qui inclut sa propre initialisation [*Linde*80]. Cet algorithme, que nous nommerons LBG, consiste à faire tourner les "k-means" en augmentant progressivement la valeur de Δ. A chaque étape, Δ est multiplié par 2 et les prototypes sont perturbés par un vecteur $\vec{P} = [\mu\mu...\mu]^T$, ou μ est un réel positif proche de zéro (d'où le nom d'initialisation par "splitting", souvent employé).

1. $\Delta = 1$, $\vec{W}_1(0) = barycentre(\mathcal{A})$, t=0
2. t=t+1
   Pour j=1 à Δ faire:
   $$\vec{W}_{2j}(t) = \vec{W}_j(t-1) + \vec{P}$$
   $$\vec{W}_{2j-1}(t) = \vec{W}_j(t-1) - \vec{P}$$
   Multiplier Δ par 2
3. Faire tourner les k-means jusqu'à convergence
4. Tant que Δ n'a pas atteint la valeur souhaitée, aller en (2)

4.2 Lien avec l'algorithme de Kohonen

Pour accélérer l'apprentissage, une technique possible consiste à réduire progressivement l'étendue du voisinage dans le modèle de Kohonen. Nous montrons ci-dessous que, durant les dernières itérations de l'apprentissage (quand l'étendue du voisinage est quasiment nulle), l'état d'équilibre du réseau est défini par les mêmes conditions que pour l'algorithme des "k-means". Toutefois, il semble

que les relations de voisinage qui ont été actives durant la plus grande partie de l'apprentissage réduisent la probabilité de blocage dans un minimum local.

A l'équilibre, en conservant les notations du paragraphe relatif à l'algorithme de Kohonen, on a:

$$E\{\Delta\vec{W}_j\} = \vec{0}$$

Donc, conformément à l'équation d'apprentissage $\Delta\vec{W}_j = \alpha(\vec{x} - \vec{W}_j)$, quand l'étendue du voisinage est nulle, la condition d'équilibre s'écrit:

$$\vec{W}_j = E\{\vec{x} \mid \forall k,\ \|\vec{x} - \vec{W}_j\|^2 < \|\vec{x} - \vec{W}_k\|^2\}$$

Par conséquent, chaque vecteur $\vec{W}_j$ est le centre de gravité des vecteurs pour lesquels $\vec{W}_j$ est le plus proche prototype. Ceci signifie que l'état d'équilibre est conditionné par les mêmes équations que pour la quantification vectorielle classique. Mais, ceci ne signifie pas que les états d'équilibres sont les mêmes, car il peut exister plusieurs états vérifiant ces équations.

Cette similitude entre les 2 algorithmes n'est valable que lorsque l'étendue du voisinage dans le modèle de Kohonen est nulle.

Annexe 1 : Méthode de Hebb et Pseudo-Inverse

La méthode de Hebb et la Pseudo-Inverse sont des classifieurs linéaires, caractérisés par une matrice W. A un vecteur $\vec{x}$, ils associent $\vec{y} = W\vec{x}$. Si $\vec{x}$ appartient à la classe k, on souhaiterait avoir $y_k = +1$ et $y_j = -1$ pour $j \neq k$. En pratique, si y_k est la composante de $\vec{y}$ qui a la plus grande valeur, $\vec{x}$ sera affecté à la classe k.

Soient $\{\vec{x}(1)...\vec{x}(N)\}$ les exemples d'apprentissage, et $\{\vec{y}(1)...\vec{y}(N)\}$ les sorties souhaitées correspondantes. Notons X et Y les matrices suivantes:

$$X = [\vec{x}(1)...\vec{x}(N)]$$
$$Y = [\vec{y}(1)...\vec{y}(N)]$$

Pseudo-Inverse:

On obtiendra exactement les sorties souhaitées si la matrice W est telle que:

$$Y = WX$$

La solution est donnée par:

$$W = YX^+$$

où X^+ est la Pseudo-Inverse de X. Si les colonnes de X sont linéairement indépendantes, on a:

$$X^+ = (X^T X)^{-1} X^T$$

Dans le cas contraire, on prend pour W la matrice qui minimise $\|Y - WX\|$. Le calcul peut se faire par un algorithme de gradient. La différence avec le réseau à 2 couches est l'absence de la tangente hyperbolique en sortie.

Méthode de Hebb:

Avec les notations présentes, la loi d'évolution décrite par Hebb est (cf chap. 1):

$$\frac{dW_{ij}}{dt} = \alpha x_i y_j - \beta W_{ij}$$

L'équilibre est atteint pour:

$$W_{ij} = \frac{\alpha}{\beta} E\{x_i y_j\}$$

Comme on peut choisir α et β, et que l'on a ici un nombre fini d'exemples, on écrira:

$$W_{ij} = \frac{1}{N} \sum_{n=1}^{N} x_i(n) y_j(n)$$

D'où:

$$\begin{aligned} W &= \frac{1}{N} \sum_{n=1}^{N} \vec{y}(n) \vec{x}^T(n) \\ &= \frac{1}{N} Y X^T \end{aligned}$$

Références

[*Arquès*82] Pierre-Yves ARQUES
"Décision en traitement du signal"
2^e édition, MASSON, 1982

[*Gray*84] R.M.GRAY
"Vector Quantization"
IEEE ASSP Magazine, April 1984

[*Kohonen*84] T.KOHONEN
"Self-Organization and Associative Memory"
Springer-Verlag, 1984

[*Linde*80] Y.LINDE, A.BUZO, R.M.GRAY
IEEE Trans. on Communications
vol COM-28, $n°1$, January 1980

[*Lippmann*88] Richard P. LIPPMANN
"Neural nets for computing"
IEEE conference on Acoustic and Speech Processing, 1988

[*Lippmann*89] Richard P. LIPPMANN
"Pattern classification using neural networks"
IEEE Communications Magazine, November 1989

[*MacQueen*67] J. Mac QUEEN
"Some methods for classification and analysis of multivariate observations"
Proc. of the fifth Berkeley Symposium on Math., Stat. and Prob.
Vol 1, pp 281-296, 1967

[*Rumelhart*86] D.E. RUMELHART, G.E. HINTON, R.J. WILLIAMS
"Learning internal representations by error backpropagation"
Parallel Distributed Processing, D.E. RUMELHART and J.L. Mc CLELLAND
Chap8, Bradford book - MIT Press - 1986

Chapitre 3:

COMPLEMENTS ALGORITHMIQUES

Nous proposons dans ce chapitre des améliorations de l'algorithme de rétropropagation du gradient, qui visent à permettre une mise en œuvre aisée et efficace sur nos applications de traitement de l'image et du signal.

Pour cela, nous développons des méthodes de réglage automatique des paramètres d'apprentissage (paramètres d'initialisation des poids synaptiques, vitesse d'apprentissage, filtrage passe-bas, etc). Nous proposons également une fonction d'erreur spécialement adaptée aux problèmes de classification. De plus, afin de limiter l'effet des phénomènes de nature à perturber l'apprentissage, un modèle de neurone qui comprend des contrôleurs de saturation et de perturbation sur les potentiels est élaboré.

1 Introduction

Nos premières expérimentations sur des problèmes de traitement d'image ont montré que la mise en œuvre de l'algorithme classique de rétropropagation est difficile, voire impossible, dès que l'on ne se limite plus à des cas d'école. Un problème important est le réglage des divers paramètres de l'algorithme (vitesse d'apprentissage, paramètres d'initialisation des poids synaptiques, ...). Un second problème provient de l'influence des phénomènes de saturation et de variations parfois trop rapides des poids. Enfin, l'erreur quadratique moyenne n'est pas adaptée lorsque le réseau de neurones est utilisé en classifieur. Par conséquent, nous avons été amenés à développer des améliorations de l'algorithme classique afin d'obtenir une convergence correcte.

L'algorithme d'apprentissage classique [*Rumelhart*86] présente l'inconvénient de nécessiter le réglage de nombreux paramètres (vitesse d'apprentissage, paramètres relatifs à l'initialisation des poids synaptiques, etc). Pour bon nombre d'applications, l'expérience montre qu'un apprentissage correct n'est obtenu que pour certaines valeurs bien précises de ces paramètres. Ceci nécessite de nombreux essais, et quand on sait qu'un apprentissage peut demander plusieurs heures de calcul, on comprend mieux l'intérêt d'un algorithme non paramétré. Nous proposons ci-dessous des méthodes permettant un réglage automatique de ces paramètres.

L'analyse de l'évolution de l'état du réseau en cours d'apprentissage montre que deux facteurs sont souvent la cause d'un blocage de l'apprentissage: les phénomènes de saturation, qui font chuter le gradient, et les perturbations induites sur les potentiels des neurones par les variations des poids. Pour résoudre ces problèmes, nous proposons un nouveau modèle de neurone (fig 1). Ce modèle contient des modules de contrôle de la saturation et de limitation des perturbations induites par les variations des poids [*Burel*90]. Nous proposons également une fonction d'erreur spécialement adaptée aux problèmes de classification. L'utilisation de l'erreur quadratique moyenne pose en effet quelques problèmes lorsque le réseau est utilisé à des fins de classification.

Cet algorithme amélioré a été utilisé avec succès pour toutes les applications décrites dans les chapitres suivants. Comme le réglage des paramètres est automatique, l'apprentissage a toujours aboutit dès le premier essai. De plus l'apprentissage est bien plus régulier qu'avec l'algorithme classique (décroissance régulière de l'erreur), et la généralisation obtenue est souvent meilleure (c'est probablement une conséquence de la régularité de l'apprentissage, qui évite au réseau d'arriver dans des états trop particuliers).

Comment évaluer l'intérêt de ces améliorations? Notons en premier lieu qu'il

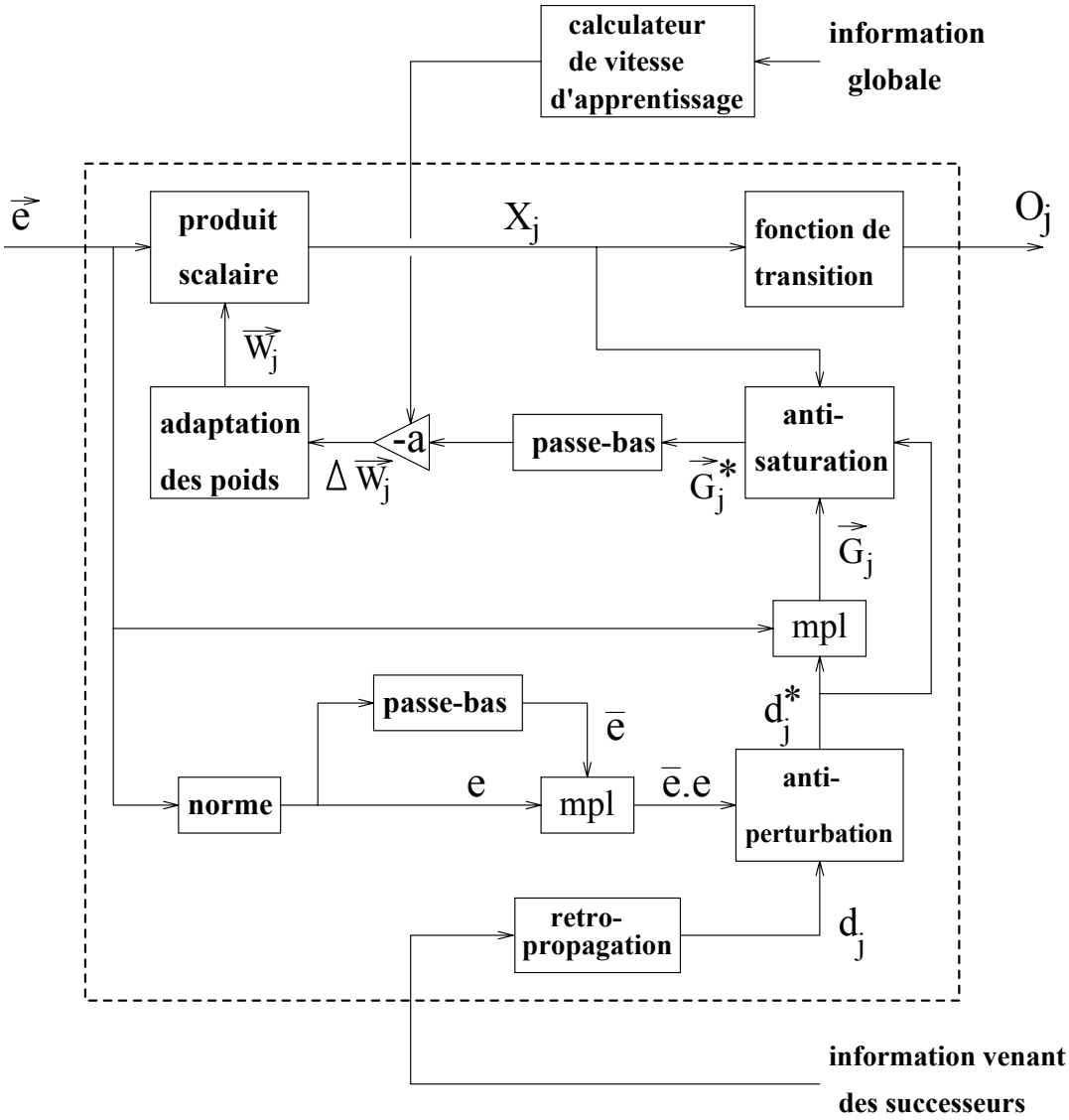


Figure 1: Nouveau modèle de neurone

est difficile d'isoler l'apport de chacune de ces améliorations; c'est pourquoi nous ne présenterons pas de tableaux comparatifs. Il existe en effet des dépendances complexes entre elles, et de tels tableaux seraient donc sans grand intérêt. Ainsi, par exemple, lorsque les poids sont correctement initialisés, et les paramètres d'apprentissage bien réglés, le contrôle des perturbations est moins sollicité.

A première vue, l'intérêt des diverses améliorations citées peut être jugé en

fonction des critères suivants:

- Gain en nombre d'essais nécessaires pour obtenir un apprentissage correct.
- A nombre d'essais égaux, gain en temps d'apprentissage.

Pour une application de difficulté moyenne (la notion de difficulté s'entendant ici au sens de "difficulté de l'apprentissage"), telle que la reconnaissance de chiffres manuscrits, une convergence satisfaisante a pu être obtenue avec l'algorithme de base, au prix d'une dizaine d'essais (10 réglages de paramètres). Pour un réglage correct des paramètres, la vitesse de convergence de l'algorithme de base est seulement de 20% inférieure à celle de notre algorithme amélioré. Par contre, pour une application très difficile telle que la séparation de sources, il n'a jamais été possible d'obtenir une convergence avec l'algorithme de base. Ceci est probablement dû à de fortes variations de l'intensité du gradient en cours d'apprentissage (des variations d'un facteur 1000 en quelques itérations ont été constatées). Il est alors impossible de converger sans une adaptation permanente de la vitesse d'apprentissage et un contrôle des perturbations induites sur les potentiels.

Signalons enfin que le fait de disposer d'un algorithme non-paramétré est important pour les applications opérationnelles, car l'opérateur ne sera pas nécessairement un spécialiste des réseaux de neurones (et il n'est donc pas question de lui demander de régler des paramètres).

2 Réglage automatique des paramètres

2.1 Initialisation des poids synaptiques

L'algorithme de rétropropagation nécessite une initialisation aléatoire des poids synaptiques de façon à briser la symétrie du réseau. La loi d'initialisation choisie est généralement une loi stochastique uniforme entre des bornes $-M_1$ et $+M_2$, comme indiqué sur la figure 2 (ces bornes ne sont pas nécessairement les mêmes pour tous les neurones). Toutefois, le choix de ces limites n'est pas aisé et de nombreux essais peuvent être nécessaires avant d'obtenir une initialisation correcte. Aussi, nous proposons ci-dessous une méthode de calcul automatique de ces valeurs.

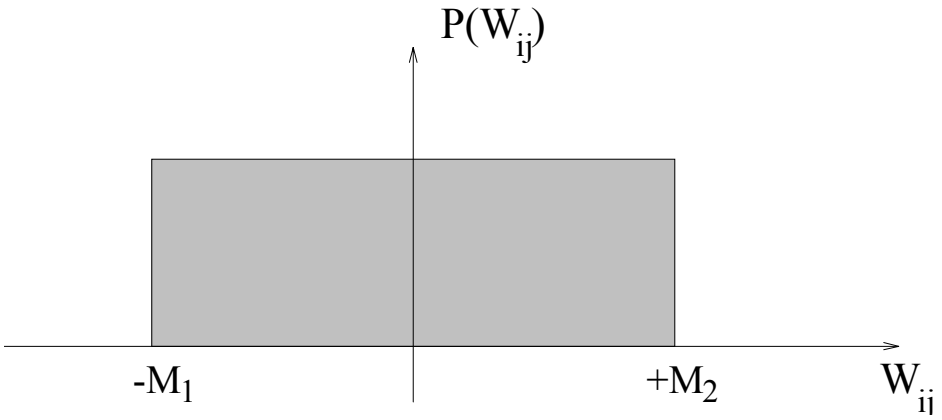


Figure 2: Initialisation des poids synaptiques

Nous allons chercher à obtenir les conditions suivantes:

$$\begin{array}{lll} (C1) & \forall j & E(X_j) = 0 \\ (C2) & \forall j & var(X_j) = \sigma_x^2 \end{array}$$

E désigne l'espérance mathématique (la moyenne est effectuée ici sur tous les exemples et tous les tirages de poids). L'objet de la première condition est simplement d'éviter un biais sur les potentiels. La seconde condition permet un contrôle de la dispersion des potentiels initiaux (la valeur σ_x sera fixée ultérieurement). Une forte dispersion conduirait à une importante différenciation initiale des neurones, d'où un apprentissage difficile. En effet, les neurones dont les poids initiaux ne seraient pas adaptés à l'application resteraient inutilisés durant les premières étapes de l'apprentissage, et un grand nombre de neurones satureraient, d'où un gradient faible et un apprentissage très lent.

La première condition conduit à :

$$\forall i, j \qquad E(W_{ij}) = 0 \tag{1}$$

En effet, comme W_{ij} est indépendant de O_i nous avons $E(X_j) = E\left(\sum_i W_{ij} O_i\right) = \sum_i E(W_{ij}) E(O_i)$. La solution la plus simple pour annuler cette somme consiste à initialiser les poids en respectant (1).

La seconde condition conduit à :

$$\forall i,j \qquad \sum_i E\{W_{ij}^2\}E\{O_i^2\} = \sigma_x^2 \tag{2}$$

En effet, on a:

$$\begin{aligned}
var(X_j) &= E\{X_j^2\} \\
&\quad \textit{car } X_j \textit{ est centré d'après } (C1) \\
&= E\{(\textstyle\sum_i W_{ij}O_i)^2\} \\
&\quad \textit{d'après la définition de } X_j \\
&= \textstyle\sum_i E\{W_{ij}^2O_i^2\} + \sum_{k\neq l} E\{W_{kj}O_kW_{lj}O_l\} \\
&= \textstyle\sum_i E\{W_{ij}^2\}E\{O_i^2\} + \sum_{k\neq l} E\{W_{kj}\}E\{W_{lj}\}E\{O_kO_l\} \\
&\quad \textit{car les poids sont indépendants entre eux et indépendants des } O_m \\
&= \textstyle\sum_i E\{W_{ij}^2\}E\{O_i^2\} \\
&\quad \textit{car les poids sont centrés d'après } (1)
\end{aligned}$$

Si les poids W_{ij} d'un neurone j sont initialisés de la même façon (pour j fixé), l'équation (2) conduit à:

$$E(W_{ij}^2) = \frac{\sigma_x^2}{\sum_i E(O_i^2)} \tag{3}$$

Appliquons à présent ces résultats à une densité de probabilité uniforme:

$$\begin{aligned}
(1) \quad &\Longrightarrow M_1 = M_2 = M \\
&\textit{et} \quad E(W_{ij}^2) = \frac{M^2}{3}
\end{aligned}$$

d'où (d'après (3)) :

$$\mathbf{M = \sqrt{\frac{3\sigma_x^2}{\sum_i E\{O_i^2\}}}}$$

$\sigma_x = \mathbf{0.5}$ semble être une bonne valeur car elle permet de se situer initialement dans la zone quasi-linéaire de la tangente hyperbolique, où le gradient est fort (fig 3). Par conséquent:

$$\mathbf{M = \frac{0.87}{\sqrt{\sum_i E(O_i^2)}}}$$

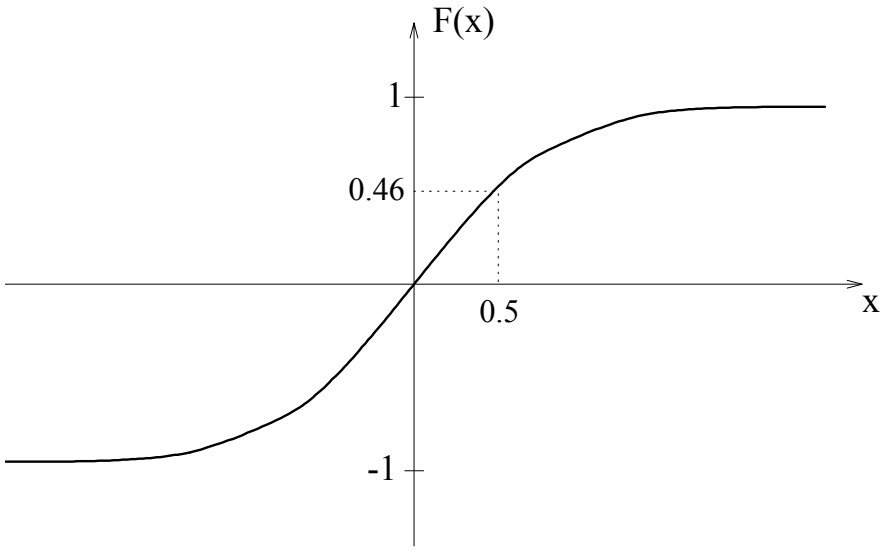


Figure 3: Fonction de transition du neurone (tangente hyperbolique)

Lorsque les entrées du neurone ont même statistiques du second ordre, on peut écrire (en notant n le nombre de prédécesseurs du neurone):

$$\mathbf{M} = \frac{\eta}{\sqrt{\mathbf{n}}}$$

avec

$$\eta = \frac{\mathbf{0.87}}{\sqrt{\mathbf{E(O_i^2)}}}$$

Exemple 1 : O_i est un élément de {-1,+1}. Ceci est le cas pour les neurones de la première couche cachée quand le réseau est alimenté par une image binaire, ou pour les neurones d'une couche supérieure quand la couche précédente sature.
Alors $E(O_i^2) = 1.0$ et $\eta = \mathbf{0.87}$

Exemple 2 : O_i est uniformément distribué dans $[-1, +1]$. Ceci est le cas pour les neurones de la première couche cachée quand le réseau est alimenté par une image normalisée (niveaux de gris ramenés entre -1 et +1) et avec un histogramme plat.
Alors $E\{O_i^2\} = \frac{1}{3}$ et $\eta = \mathbf{1.50}$

Notre logiciel calcule η pour les neurones de la première couche en effectuant des statistiques sur les exemples (calcul des $E\{O_i^2\}$). Pour les autres couches il

calcule η comme si les états de la couche précédente étaient uniformément distribués dans $[-1, +1]$.

2.2 Facteur d'oubli du filtrage passe-bas

[Les vecteurs sont notés par des minuscules et leurs normes par des majuscules]

Le gradient est filtré par un passe-bas du premier ordre selon:

$$\bar{g}(t) = (1 - \beta)g(t) + \beta\bar{g}(t - 1) \tag{4}$$

Le choix du facteur d'oubli β est laissé à l'opérateur. Bien que sa valeur ne soit pas cruciale, on remarque toutefois que des valeurs mal choisies peuvent parfois ralentir l'apprentissage. Les expérimentations suggèrent par exemple qu'il est souhaitable de prendre des valeurs plus faibles de β lorsque l'on a peu d'exemples, mais il n'est pas possible d'en tirer une loi empirique suffisamment générale. C'est pourquoi nous proposons ci-dessous une méthode qui permet au système de calculer automatiquement la valeur de β.

Supposons que les poids varient assez lentement en cours d'apprentissage (ce qui est généralement le cas en pratique, à moins que l'on ne choisisse une vitesse d'apprentissage trop importante). Pour un certain jeu de poids, notons g_0 le vecteur gradient moyen (moyenne sur tous les exemples). Comme les exemples sont présentés dans un ordre aléatoire, le processus est ergodique et on a $g_0 = E\{g(t)\}$.
Si le filtrage passe-bas est efficace, $\bar{g}$ doit être proche de g_0. Comme les exemples sont présentés dans un ordre aléatoire, $\bar{g}$ est un vecteur stochastique. Donc, "proche de" s'exprime mathématiquement par "$E\{\|\bar{g} - g_0\|^2\}$ est faible devant ${G_0}^2$":

$$E\{\|\bar{g} - g_0\|^2\} = \epsilon^2 {G_0}^2 \tag{5}$$

où ϵ est une valeur faible devant 1. Nous prendrons $\epsilon^2 = 0.1$, car il n'est pas souhaitable de prendre une valeur trop faible (par exemple 0.01). Le maintient d'une certaine variance en sortie du filtre est en effet bénéfique car cela permet d'échapper à des minima locaux en bruitant l'apprentissage. Quant au vecteur g_0, il sera estimé par une moyenne sur tous les exemples.

Calculons cette variance en posant $g(t) = g_0 + b(t)$. D'abord, on vérifiera aisément que l'on a $E\{\bar{g}(t)\} = g_0$ (il suffit d'appliquer l'espérance sur l'équation (4) et de se souvenir du fait que $g_0 = E\{g(t)\}$).

L'équation (4) peut s'écrire:

$$\bar{g}(t) - g_0 = (1-\beta)\,(g(t) - g_0) + \beta\,(\bar{g}(t-1) - g_0)$$

soit encore:

$$\bar{g}(t) - g_0 = (1-\beta)b(t) + \beta\,(\bar{g}(t-1) - g_0)$$

d'où:

$$\|\bar{g} - g_0\|^2 = (1-\beta)^2 B^2(t) + \beta^2 \|\bar{g}(t-1) - g_0\|^2 + 2\beta(1-\beta)b(t).\,(\bar{g}(t-1) - g_0)$$

Comme les exemples sont présentés aléatoirement, b(t) est indépendant de $\bar{g}(t-1)$. Si l'on veut être parfaitement rigoureux, ceci n'est pas tout à fait exact car comme chaque exemple est tiré une seule fois sur un passage on sait que les exemples tirés jusqu'à t-1 ne le seront plus jusqu'à la fin du passage complet. Toutefois, la dépendence qui en découle entre b(t) et $\bar{g}(t-1)$ est probablement faible. En tenir compte conduirait à une importante complication des calculs pour un gain de précision négligeable sur β. Donc, l'expression précédente conduit à:

$$E\{\|\bar{g}(t) - g_0\|^2\} = (1-\beta)^2 E\{B^2(t)\} + \beta^2 E\{\|\bar{g}(t-1) - g_0\|^2\}$$

Les propriétés statistiques de $\bar{g}$ varient lentement, d'où:

$$E\{\|\bar{g} - g_0\|^2\} = \frac{(1-\beta)^2}{1-\beta^2} E\{B^2\}$$

Et par identification avec (5) on obtient:

$$\frac{(1-\beta)^2}{1-\beta^2} = \frac{\epsilon^2 G_0^2}{E\{B^2\}}$$

La résolution conduit à:

$$\beta = \frac{\mathbf{1} - \mathbf{k^2}}{\mathbf{1} + \mathbf{k^2}}$$

Avec:

$$k = \frac{\epsilon G_0}{\sqrt{E\{B^2\}}}$$

Il est toutefois souhaitable de limiter la valeur supérieure de β sur un critère de temps de réponse:

Si l'on présente en entrée du passe-bas un échelon unité sur une des composantes, la réponse est:

$$1-\beta,\ (1-\beta)(1+\beta),\ (1-\beta)(1+\beta+\beta^2),\ ...,\ 1-\beta^{n+1},\ ...$$

On atteint 90% de la valeur finale après un nombre de périodes d'échantillonnage égal à:

$$n = \frac{Ln(0.1)}{Ln(\beta)} - 1$$

Si l'on impose (en notant N_e le nombre d'exemples):

$$n \leq \frac{N_e}{2}$$

on obtient la condition:

$$\beta \leq exp\left\{\frac{Ln(0.1)}{\frac{N_e}{2}+1}\right\}$$

Il est intéressant de remarquer que cette contrainte rejoint l'habitude selon laquelle on prend de plus faibles valeurs de β lorsque le nombre d'exemples est faible.

RESUME:

$$\beta = \frac{\mathbf{1-k^2}}{\mathbf{1+k^2}}$$

Avec:

$$k = \frac{\epsilon G_0}{\sqrt{E\{B^2\}}}$$

Sous la contrainte:

$$0 \leq \beta \leq exp\left\{\frac{Ln(0.1)}{\frac{N_e}{2}+1}\right\}$$

k	0.010	0.018	0.032	0.056	0.10	0.18	0.32	0.56	1.0
β	0.9999	0.9994	0.998	0.994	0.980	0.938	0.82	0.52	0.0

N_e	4	10	30	100	300	1000	3000	10000
β_{max}	0.460	0.680	0.870	0.960	0.980	0.995	0.998	0.999

Il n'est pas utile de calculer β en continu pendant l'apprentissage. D'une part la valeur précise de β n'est pas fondamentale, et comme les poids ne varient pas trop vite, la valeur trouvée varierait très lentement. La stratégie adoptée consiste à recalculer β seulement tous les 20 passages de tous les exemples. On fait alors un passage sans modification des poids pendant lequel on calcule G_0 et $E\{B^2\}$. De ces valeurs, on déduit aisément β à l'aide des formules précédentes.

D'un point de vue qualitatif, le réglage automatique proposé réduit la force du filtrage passe-bas lorsque la variance en sortie du filtre chute. Ceci permet de maintenir une certaine "agitation", qui est bénéfique pour l'évitement des minima locaux. Par contre, lorsque cette variance devient trop importante, le réglage automatique renforce le filtrage, afin d'éviter une déstabilisation de l'apprentissage.

Signalons enfin que pour les application décrites dans les chapitres ultérieurs, la valeur trouvée pour β est généralement de l'ordre de 0.98

2.3 Vitesse d'apprentissage

[On utilise les même notations et conventions d'écriture que dans le paragraphe précédent]

Nous avons vu que la valeur du facteur d'oubli du filtrage passe-bas n'est pas cruciale, et une valeur de l'ordre 0.98 semble bien adaptée pour les applications que nous avons développées. Le problème est très différent pour ce qui est du choix de la

vitesse d'apprentissage α. Ceci n'est pas une tâche aisée, même pour le spécialiste, et de nombreux essais sont souvent nécessaires avant de trouver une bonne valeur. Des valeurs trop fortes de α conduisent à des apprentissages très irréguliers. De fortes oscillations de l'erreur sont observées, et, ensuite, le réseau est généralement piégé dans un minimum local. D'un autre coté, des valeurs trop faibles de la vitesse d'apprentissage conduisent à un apprentissage régulier, mais extrèmement lent.

La valeur optimale de la vitesse d'apprentissage est très dépendante de la taille du réseau et des exemples constituant l'ensemble d'apprentissage. Afin de surmonter ce problème, l'idée est de se ramener à un paramètre dont la valeur optimale est peu dépendante de l'application, et d'être capable de déduire α de ce paramètre. Un paramètre robuste est la décroissance relative (k_0) souhaitée sur l'erreur à chaque passage de tous les exemple: ($\Delta e_{QM} = -k_0 e_{QM}$). De cette valeur k_0 on peut aisément déduire la décroissance moyenne dûe à un exemple : $k = k_0/(nombre\ d'exemples)$.

Notons w le vecteur contenant tous les poids W_{ij} du réseau, $\bar{g}$ le gradient filtré, et g_0 le gradient moyen. Lorsqu'un exemple est présenté en entrée du réseau, la variation de poids résultante est en moyenne:

$$\begin{aligned} E\{\Delta w\} &= E\{-\alpha \bar{g}\} \\ &= -\alpha g_0 \end{aligned}$$

Donc, la décroissance de l'erreur résultante est en moyenne:

$$\begin{aligned} \Delta e_{QM} &= g_0.\Delta w \\ &= -\alpha \|g_0\|^2 \end{aligned}$$

Par conséquent, la valeur de α qui réalisera la décroissance souhaitée sera:

$$\alpha = \frac{k.e_{QM}}{\|g_0\|^2} \tag{6}$$

Notre logiciel utilise cette expression pour ajuster automatiquement le paramètre α en cours d'apprentissage. α est mis à jour à chaque passage de tous les exemples. La valeur de k_0 a été 0.05 pour toutes nos applications (sauf en séparation de sources: voir chapitre 10). Cette valeur semblant donner de bons résultats dans diverses conditions, elle constitue le réglage par défaut réalisé par notre logiciel d'expérimentation.

Toutefois, l'expression ci dessus pour le calcul de α peut conduire à des problèmes vers la fin de l'apprentissage, car le gradient tend alors vers 0 du fait que l'on approche d'un minimum. Une astuce qui fonctionne très bien en pratique consiste à remplacer $\|g_0\|^2$ par $E\{\|\bar{g}\|^2\}$ dans l'équation 6:

$$\alpha = \frac{k.e_{QM}}{E\{\|\bar{g}\|^2\}} \tag{7}$$

On peut vérifier que l'on a:

$$E\{\|\bar{g}\|^2\} = E\{\|\bar{g} - g_0\|^2\} + \|g_0\|^2$$

Lorsque le filtrage passe-bas (cf paragraphe précédent) n'est pas limité par le critère de temps de réponse (ce qui est généralement le cas en début d'apprentissage), on a:

$$E\{\|\bar{g}\|^2\} = (1 + \epsilon^2)\|g_0\|^2$$

On a donc une bonne approximation car ϵ^2 est faible ($\epsilon^2 = 0.1$). Par contre, vers la fin de l'apprentissage, c'est généralement le critère de temps de réponse qui limite la puissance du filtrage passe-bas. Il subsiste donc une variance non-nulle en sortie du filtre (c.a.d. que $E\{\|\bar{g} - g_0\|^2\}$ ne tend pas vers 0). En conséquence, $E\{\|\bar{g}\|^2\}$ ne tend pas vers 0.

$E\{\|\bar{g}\|^2\}$ est calculé exactement, par une moyenne sur tous les exemples. D'autre part, nous limitons la valeur que peut prendre α à une excursion de 4 ordres de grandeur par rapport à la valeur en début d'apprentissage. Mais en pratique, cette limitation n'a jamais été sollicitée, du fait des remarques précédentes.

Cette méthode très simple pour le calcul de la vitesse d'apprentissage est moins coûteuse (mais aussi moins rigoureuse), que des méthodes du second ordre, qui nécessiteraient le calcul du Hessien.

3 Contrôle des facteurs déstabilisants

3.1 Contrôle de la saturation

L'analyse des états des neurones des couches cachées durant des apprentissages difficiles montre qu'ils se trouvent souvent dans des états fortements saturés. Aussi, l'apprentissage est considérablement ralenti car le gradient en amont d'un neurone saturé est extrèmement faible (il suffit d'observer la figure 3 pour s'en rendre compte). Comme $\frac{\partial E}{\partial W_{ij}} = \delta_j O_i$ avec $\delta_j = (O_j - S_j)F'(X_j)$ sur la couche de sortie, et $\delta_j = \left(\sum_{k \in succ(j)} \delta_k W_{jk}\right) F'(X_j)$ sur les autres couches, il est aisé de vérifier que la saturation signifie de très faibles valeurs de $F'(X_j)$, et par conséquent de très faibles valeurs du gradient.

Il serait souhaitable d'interdire des modifications de poids tendant à augmenter $|X_j|$ trop fortement. Considérons la figure 1 . Sans le module Anti-Saturation nous aurions $\vec{G}_j^* = \vec{G}_j$. Lorsqu'un vecteur $\vec{e}$ est présenté en entrée du neurone j, il contribue à la variation courante des poids avec un coefficient multiplicatif $(1-\beta)$. Mais il contribuera également aux variations futures avec des coefficients multiplicatifs $(1-\beta)\beta, (1-\beta)\beta^2, ...$, du fait de l'effet de mémoire du filtre passe-bas. Nous noterons à présent $\vec{W}_j$ le vecteur qui contient tous les poids du neurone, et $\Delta\vec{W}_j^\infty$ la contribution totale de $\vec{e}$ à la variation de ce vecteur poids. Nous avons:

$$\begin{aligned}\Delta\vec{W}_j^\infty &= \alpha(1-\beta)(1+\beta+\beta^2+...)\delta_j^*\vec{e} \\ &= \alpha\delta_j^*\vec{e}\end{aligned}$$

L'influence de $\Delta\vec{W}_j^\infty$ sur le potentiel correspondant à $\vec{e}$ est:

$$\begin{aligned}\Delta X_j &= \Delta\vec{W}_j^\infty.\vec{e} \\ &= \alpha\delta_j^* e^2\end{aligned}$$

La saturation tend à croître quand $X_j\Delta X_j > 0$ (ce qui est équivalent à $X_j\delta_j^* > 0$). Si nous tolérons des valeurs de $|X_j|$ inférieures à 3.0 (ce qui correspond à une variation d'un facteur 100 sur la dérivée: $F'(3.0)/F'(0.0) = 0.01$), la fonction qui doit être réalisée par le module Anti-Saturation est:

$$\begin{gathered}\textbf{SI } (|X_j| \geq 3.0) \textbf{ et } (X_j\delta_j^* > 0) \\ \textbf{ALORS } \vec{G}_j^* = \vec{0} \\ \textbf{SINON } \vec{G}_j^* = \vec{G}_j\end{gathered}$$

Ce module très simple conduit pour certaines applications à des résultats remarquables, en évitant un blocage de l'apprentissage dans des zones difficiles.

3.2 Limitation de la perturbation sur les potentiels

Pour des raisons de simplicité de la démonstration, nous ne prenons pas en compte ci-dessous l'influence du module Anti-Saturation (on suppose $\vec{G}_j^* = \vec{G}_j$). On peut vérifier que cela n'affecte pas le résultat final.

Comme nous l'avons vu au paragraphe précédent, la modification des poids dûe à la présentation d'un vecteur d'entrée $\vec{e}$ (vecteur d'entrée du neurone) est (sur le long terme):

$$\Delta\vec{W}_j^\infty = \alpha\vec{G}_j = \alpha\delta_j^*\vec{e}$$

L'influence de cette perturbation sur le potentiel correspondant à un autre vecteur d'entrée $\vec{e}\,'$ est:

$$\Delta X'_j = \Delta \vec{W}_j^{\infty}.\vec{e}\,' = \alpha \delta_j^* \vec{e}.\vec{e}\,'$$

Le pire des cas a lieu lorsque $\vec{e}\,'$ est parallèle à $\vec{e}$:

$$|\Delta X'_j| = \alpha \delta_j^* e e'$$

Nous pouvons estimer $|\Delta X'_j|$ en remplacant e' par $\bar{e}$, qui est la valeur moyenne de la norme des vecteurs d'entrée obtenue par filtrage passe bas:

$$\bar{e}(t) = (1-\beta)e(t) + \beta \bar{e}(t-1)$$

De façon à limiter l'influence de $\Delta \vec{W}_j$ sur le potentiel correspondant à une autre entrée, nous imposons à $|\Delta X'_j|$ d'être inférieur à 1% de l'amplitude crête à crête tolérée sur le potentiel (cf paragraphe précédent):

$$|\Delta X'_j|_{MAX} = 0.06$$

Par conséquent, la fonction qui doit être réalisée par le module Anti-Perturbation est:

$$\delta_{\mathbf{j}}^* = \mathbf{sgn}(\delta_{\mathbf{j}}) \ \mathbf{MIN}\left(\frac{\mathbf{0.06}}{\alpha \mathbf{e}\bar{\mathbf{e}}}, |\delta_{\mathbf{j}}|\right)$$

4 Fonction d'erreur améliorée

Nous proposons dans ce paragraphe une fonction d'erreur spécialement adaptée aux problèmes de classification.

(a) LA FONCTION D'ERREUR CLASSIQUE

La fonction d'erreur généralement utilisée est l'erreur quadratique moyenne (cf chap. 2). Etudions un cas où le réseau de neurones fonctionne en classifieur (les valeurs souhaitées en sortie sont ± 1). Il est intéressant de tracer l'erreur élémentaire $E_j = \frac{1}{2}(O_j - S_j)^2$ en fonction du potentiel X_j du neurone j (fig 4).

Etudions le cas $S_j = -1$ (le raisonnement est similaire pour le cas $S_j = +1$). Lorsque X_j est très positif, l'erreur est importante, mais le gradient est faible. En conséquence, cette erreur sera corrigée très lentement par l'algorithme de rétropropagation, et l'apprentissage va être très long.

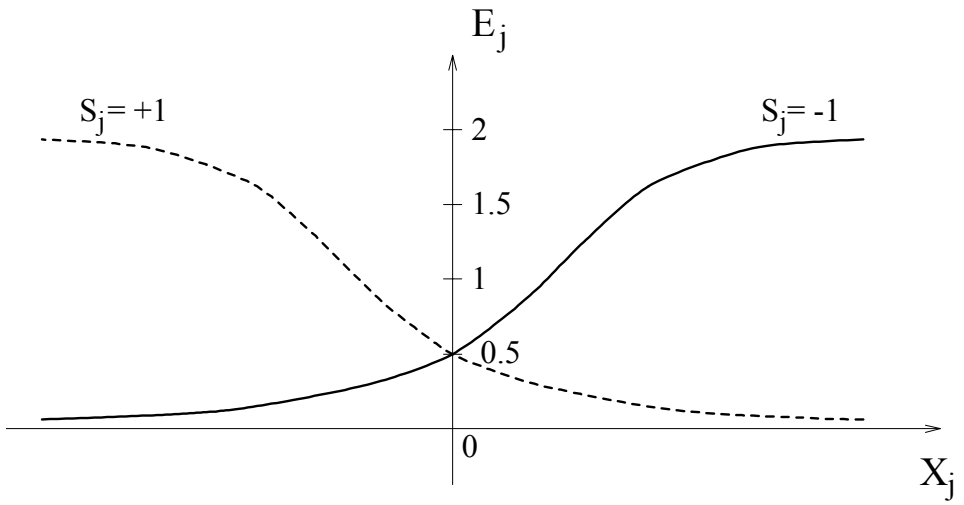


Figure 4: Erreur quadratique

Une astuce pour résoudre ce problème a été proposée dans [*Ernoult*88]. Elle consiste à prendre $F'(X_j) = F'(0)$ quel que soit X_j dans l'expression du gradient. Comme c'est la décroissance de $F'(X_j)$ pour les grandes valeurs de X_j qui est à l'origine du problème précédemment soulevé, celui-ci semble donc résolu. Mais le cadre mathématique devient flou car la fonction d'erreur minimisée est inconnue (en effet le nouveau gradient n'est plus celui de l'erreur quadratique). Nous proposons donc une approche différente, qui consiste simplement à redéfinir l'erreur à minimiser. On conserve de ce fait un cadre mathématique clair.

(b) LA FONCTION D'ERREUR PROPOSEE

Etudions la figure 4 à nouveau quand $S_j = -1$ (le raisonnement peut être fait de façon symétrique pour $S_j = +1$). Quand X_j devient de plus en plus positif, l'erreur augmente mais le gradient diminue. Donc, il serait souhaitable de redéfinir la fonction d'erreur dans cette zone, de sorte que le gradient soit une fonction croissante de l'erreur. Ceci conduit à la condition:

$$\frac{\partial^2 E_j}{\partial X_j^2} \geq 0$$

Le même résultat est obtenu lorsque l'on étudie le cas $S_j = +1$
Par conséquent, la fonction de coût doit être à courbure positive. Ceci est vérifié pour les faibles valeurs de X_j mais E_j doit être modifié pour les valeurs positives de X_j. Est il souhaitable de choisir une forte courbure? Des expériences menées avec une fonction d'erreur exponentielle nous ont montré que ce choix augmente considérablement la probabilité d'être piégé dans un minimum local. Ce résultat peut

être aisément expliqué par le fait qu'il peut être nécessaire, en cours d'apprentissage, d'augmenter temporairement l'erreur sur l'un des neurones de sortie afin de trouver un chemin vers l'optimum global. Or, une fonction d'erreur à forte courbure interdit ce type de comportement en pénalisant trop fortement les erreurs locales importantes (une erreur importante sur un seul neurone est bien plus pénalisante qu'une erreur moyenne répartie sur un grand nombre de neurones). Par conséquent, nous avons choisi une fonction à courbure nulle dans la zone où l'erreur doit être modifiée (fig 5).

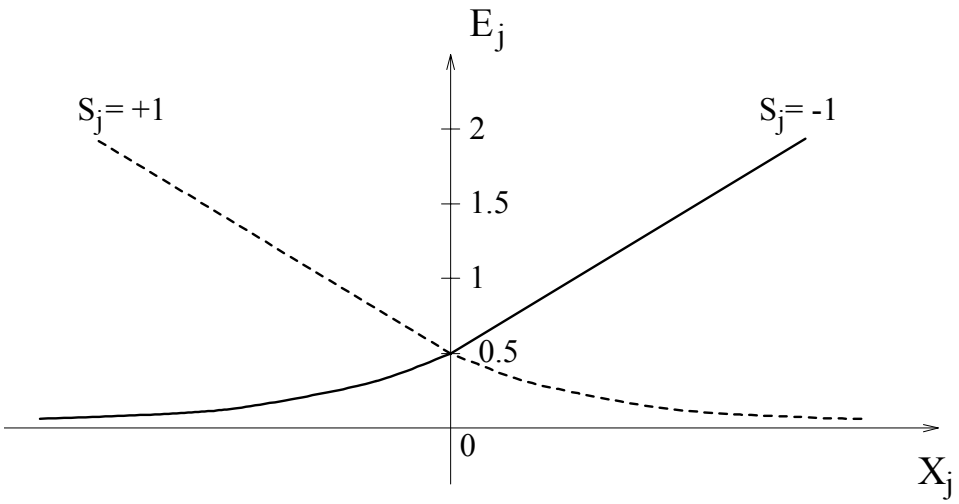


Figure 5: Fonction d'erreur proposée

Une analyse plus générale est possible sur la base des hypothèse suivantes (qui sont valables pour la majorité des fonctions non-linéaires utilisées dans le domaine des réseaux de neurones):

(H1) $F(X_j)$ est une fonction croissante : $F'(X_j) > 0$
(H2) $G(X_j) = F(X_j) - F(0)$ est une fonction antisymétrique : $G(-X_j) = -G(X_j)$
(H3) $F(\infty)$ est une valeur finie
(H4) Les valeurs désirées en sortie sont $F(\infty)$ ou $-F(\infty)$
(H5) $sgn(F''(X_j)) = -sgn(X_j)$

Avec l'erreur quadratique E_j nous avons:

$$E_j = \frac{1}{2}(F(X_j) - S_j)^2$$

$$\frac{\partial E_j}{\partial X_j} = (F(X_j) - S_j)F'(X_j)$$

$$\frac{\partial^2 E_j}{\partial {X_j}^2} = (F(X_j) - S_j)F''(X_j) + F'^2(X_j)$$

Et nous imposons une contrainte de convexité (de sorte que le gradient soit une fonction croissante de l'erreur):

$$\frac{\partial^2 E_j}{\partial X_j^2} \geq 0$$

Ceci est le cas lorsque $sgn((F(X_j) - S_j)F''(X_j)) \geq 0$, mais peut ne plus être vérifié quand cette expression est négative. En conséquence, nous modifions la fonction d'erreur dans cette zone, qui correspond (d'après H5) à $sgn((F(X_j) - S_j)X_j) > 0$. La nouvelle fonction d'erreur est :

$$E_j = aX_j + b$$

Comme $(F(X_j) - S_j)$ est de signe constant (d'après H1 et H4), la frontière est en $X_j = 0$. Donc, les coefficients a et b peuvent être aisément calculés par continuité et continuité de la dérivée en $X_j = 0$:

$$b = \frac{1}{2}(F(0) - S_j)^2$$

$$a = (F(0) - S_j)\, F'(0)$$

Et l'erreur est :

$$\begin{array}{ll} E_j = \frac{1}{2}\left(F(X_j) - S_j\right)^2 & quand\ |F(X_j) - S_j| \leq \frac{1}{2}\left(F(\infty) - F(-\infty)\right) \\ E_j = aX_j + b & quand\ |F(X_j) - S_j| > \frac{1}{2}\left(F(\infty) - F(-\infty)\right) \end{array}$$

(en utilisant (H1)(H2)(H3)).

Le tableau suivant indique les coefficients obtenus pour les fonctions non-linéaires les plus souvent utilisées.

$F(X_j)$	$F(0)$	$F'(0)$	S_j	a	b
$th(X_j)$	0.0	1.0	-1	1.0	0.5
			$+1$	-1.0	0.5
$\frac{1}{2}(1 + th(X_j))$	0.5	0.5	0.0	0.25	0.125
			1.0	-0.25	0.125

Notons que (pour les neurones de sortie) dans la zone correspondant à $|F(X_j) - S_j| > \frac{1}{2}\left(F(\infty) - F(-\infty)\right)$ nous avons $\delta_j = -a$ car $\delta_j = -\frac{\partial E_j}{\partial X_j}$

Références

[*Burel*90] Gilles BUREL, Dominique CAREL, Jean-Yves CATROS
"A connectionist system for recognition of 2D workpieces"
Revue technique Thomson CSF, vol 22, n^o 4, Décembre 1990

[*Ernoult*88] Christine ERNOULT
"Performance of back-propagation on a parallel transputer-based machine"
Congrès Neuro-Nîmes 88, Nîmes, 15-17 Novembre 1988

[*Rumelhart*86] D.E. RUMELHART, G.E. HINTON, R.J. WILLIAMS
"Learning internal representations by error backpropagation"
Parallel Distributed Processing, D.E. RUMELHART and J.L. Mc CLELLAND
Chap8, Bradford book - MIT Press - 1986

Chapitre 4:

NOUVEAUX ALGORITHMES ET RESULTATS THEORIQUES

La structure la plus largement utilisée dans le domaine des réseaux de neurones est le Perceptron Multicouches. Nous proposons un nouveau modèle de neurone compatible avec ce type de réseau. Ce neurone utilise un codage particulier de sa sortie, que nous nommerons "Représentation Scalaire Distribuée". Cette représentation accroît considérablement la capacité du réseau à traiter des problèmes non-linéaires. Après la présentation de notre modèle et l'introduction de résultats théoriques utiles à sa compréhension, nous le validerons sur une application de réduction de dimensionnalité.

Dans la seconde partie du chapitre, nous nous intéressons au modèle de Kohonen. Notre objectif est de présenter des résultats théoriques susceptibles d'aider à la compréhension et à la mise en œuvre de cet algorithme. L'attention est portée sur l'interprétation comme algorithme de minimisation d'une fonction d'énergie, et des algorithmes similaires, plus proches des algorithmes traditionnels de Quantification Vectorielle sont proposés. Enfin, des résultats expérimentaux sont présentés.

1 Une nouvelle approche pour les réseaux de neurones: la Représentation Scalaire Distribuée

1.1 Introduction

Nous proposons un nouveau modèle de neurone compatible avec la structure du perceptron multi-couches. Ce modèle accroît considérablement les capacités du réseau à traiter des problèmes non-linéaires. L'idée consiste à représenter la sortie du neurone non pas par un scalaire, mais par une fonction. Pour nommer ce passage de l'espace des scalaires à l'espace des fonctions, nous parlerons de "Représentation Scalaire Distribuée".

Nous présentons le modèle proposé, et l'algorithme d'apprentissage correspondant. Des résultats théoriques utiles à sa compréhension sont également introduits. Notre modèle sera ensuite validé sur deux applications: la réduction de dimensionnalité (dans un cas non-linéaire) et la compression d'images (prédiction de codes, chapitre 9).

1.2 La Représentation Scalaire Distribuée

Pour présenter la notion de Représentation Scalaire Distribuée, nous noterons $\mathcal{R}$ l'ensemble des réels, $\mathcal{F}$ l'ensemble des applications de $\mathcal{R}$ dans $\mathcal{R}$, et $\mathcal{D}$ l'ensemble des injections de $\mathcal{R}$ dans $\mathcal{F}$. Une injection d de $\mathcal{D}$ sera nommée "fonction de distribution":

$$\begin{array}{rccc} d: & \mathcal{R} & \longrightarrow & \mathcal{F} \\ & x & \longrightarrow & f_x \end{array}$$

Cette fonction de distribution étant injective, elle n'introduit pas de perte d'information. La fonction f_x sera nommée "Représentation Scalaire Distribuée" de x.

Nous considérerons à présent l'injection d_h, qui à tout réel x associe le résultat du filtrage de l'impulsion δ_x (impulsion de Dirac centrée en x) par une fonction paire et apériodique h:

$$\begin{array}{rccc} d_h: & \mathcal{R} & \longrightarrow & \mathcal{F} \\ & x & \longrightarrow & f_x = \delta_x * h \end{array}$$

Dans ce cas, on a $f_x(y) = h(y-x) = h(x-y)$. On vérifiera aisément que l'apériodicité de h est une condition nécessaire et suffisante pour que d_h soit injective. Nous

verrons plus loin qu'il est souhaitable pour l'efficacité de l'apprentissage que h soit une fonction de lissage.

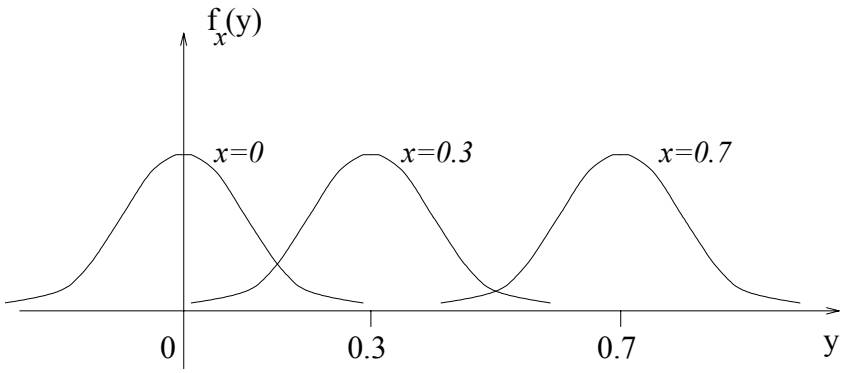


Figure 1: Représentation Scalaire Distribuée

L'opération de distribution est entièrement déterminée par la connaissance de la fonction de filtrage h. En pratique, nous prendrons pour h une fonction d'allure gaussienne. La figure 1 illustre la RSD des scalaires $x = 0,\ x = 0.3\ et\ x = 0.7$, lorsque la fonction h est une gaussienne.

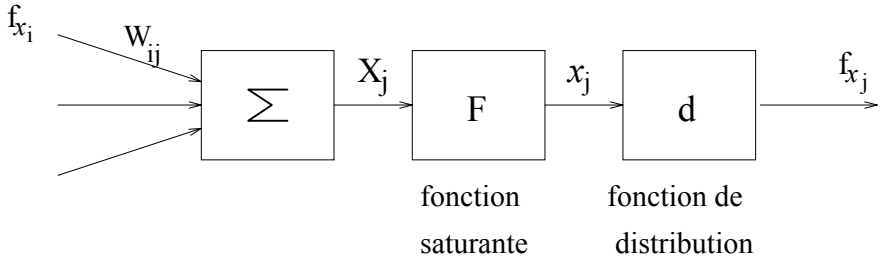


Figure 2: Modèle du neurone (en continu)

Le modèle du neurone est fourni figure 2. Il reçoit en entrée les fonctions f_{x_i} provenant des neurones i de la couche précédente, et calcule une somme pondérée de ces fonctions:

$$X_j = \sum_{i \in pred(j)} \int_{-\infty}^{+\infty} W_{ij}(u) f_{x_i}(u) du$$

W_{ij} est un continuum de poids caractérisant la connexion ij. Le résultat de cette somme pondérée passe dans une fonction saturante F dont l'intérêt apparaîtra

plus tard. Au résultat $x_j = F(X_j)$, on applique la fonction de distribution d_h, afin d'obtenir la fonction de sortie du neurone: $f_{x_j} = d_h(x_j) = \delta_{x_j} * h$.

Pour mieux comprendre le fonctionnement du réseau, il est intéressant d'étudier une connexion ij. D'après l'expression de X_j, on peut écrire:

$$X_j = \sum_{i \in pred(j)} C_{ij}(x_i)$$

avec

$$\begin{aligned} C_{ij}(x_i) &= \int_{-\infty}^{+\infty} W_{ij}(u) f_{x_i}(u) du \\ &= \int_{-\infty}^{+\infty} W_{ij}(u) h(x_i - u) du \\ &= (h * W_{ij})(x_i) \end{aligned}$$

Ainsi, en regroupant la fonction de distribution du neurone i avec le continuum de poids caractérisant la connexion ij, on peut voir la connexion ij comme une fonction $C_{ij} = h * W_{ij}$, résultat du filtrage du continuum de poids par la fonction génératrice de la RSD. On peut donc décrire une propagation dans le neurone par la succession d'opérations suivantes:

$$\begin{aligned} X_j &= \sum_{i \in pred(j)} C_{ij}(x_i) \\ x_j &= F(X_j) \end{aligned}$$

Ainsi, si l'on souhaite garder le modèle classique du neurone, on peut rendre compte de l'effet de la RSD en considérant que la connexion est une fonction adaptative, et non un simple gain adaptatif comme dans le cas du neurone classique. De plus, les neurones seuils deviennent inutiles car chaque connexion peut régler elle même son "offset".

1.3 Echantillonnage de la RSD

En pratique il n'est pas possible de représenter un continuum de poids. Les poids seront donc représentés par la fonction échantillonnée suivante (n désigne un entier, et δ est la distribution de Dirac):

$$W_{ij}(u) = \sum_{n=-\infty}^{+\infty} W_{ij,n} \delta(n - u)$$

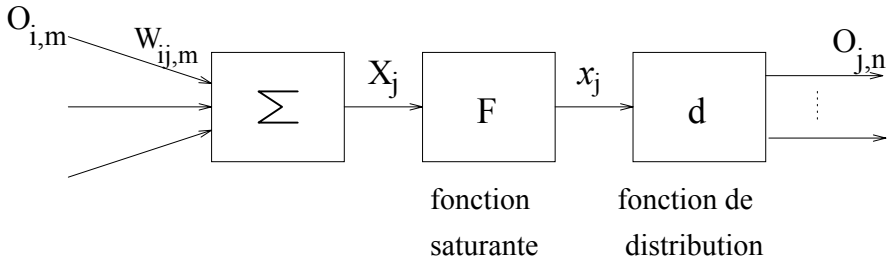


Figure 3: Modèle du neurone (en échantilloné)

On a alors :

$$\begin{aligned} C_{ij}(x_i) &= (h * W_{ij})(x_i) \\ &= \int_{-\infty}^{+\infty} \left(\sum_{n=-\infty}^{+\infty} W_{ij,n} \delta(n-u) \right) f_{x_i}(u) du \\ &= \sum_{n=-\infty}^{+\infty} W_{ij,n} \int_{-\infty}^{+\infty} \delta(n-u) f_{x_i}(u) du \\ &= \sum_{n=-\infty}^{+\infty} W_{ij,n} f_{x_i,n} \end{aligned}$$

Le modèle du neurone en échantillonné est représenté figure 3. On a posé $O_{j,n} = f_{x_j,n}$.

1.4 Limitation de la RSD à un intervalle

On fixera à présent à 0 les poids $W_{ij,n}$ pour les valeurs de n extérieures à un intervalle $[n_1, n_2]$. En pratique, les bornes de l'intervalle seront choisies de telle sorte quelles encadrent les valeurs asymptotiques de la fonction F. Ceci se justifie par le fait que le filtre h est en pratique une gaussienne d'écart type assez faible, et les poids d'indice extérieur à $[n_1, n_2]$ ont donc peu d'influence.

Notons que le cas d'un pas d'échantillonage non constant n'a pas été envisagé car on peut toujours se ramener à un pas unité par un choix convenable de la fonction F.

On a donc à présent:

$$C_{ij}(x_i) = \sum_{n=n_1}^{n_2} W_{ij,n} O_{i,n}$$

1.5 Algorithme d'apprentissage

L'algorithme d'apprentissage est un algorithme de gradient, inspiré de l'algorithme de rétropropagation.

On se définit par exemple une erreur quadratique moyenne entre les sorties obtenues (x_j) et les sorties souhaitées (t_j):

$$e_{QM} = E\{e_Q\}$$

avec

$$e_Q = \frac{1}{2} \sum_{j \in sortie} (x_j - t_j)^2$$

On présente plusieurs fois tous les exemples dans un ordre aléatoire. A chaque présentation d'un exemple, on fait varier les poids $W_{ij,n}$ dans le sens inverse du gradient de l'erreur :

$$\Delta W_{ij,n} = -\alpha \frac{\partial e_{QM}}{\partial W_{ij,n}}$$

α étant une constante positive, on s'assure ainsi une diminution de l'erreur.
Nous noterons à présent $g_{ij,n}$ le gradient local de l'erreur quadratique:

$$g_{ij,n} = \frac{\partial e_Q}{\partial W_{ij,n}}$$

Comme $e_{QM} = E\{e_Q\}$, on peut écrire:

$$\Delta W_{ij,n} = -\alpha E\{\frac{\partial e_Q}{\partial W_{ij,n}}\}$$

car la dérivation est un opérateur linéaire. On a donc :

$$\Delta W_{ij,n} = -\alpha E\{g_{ij,n}\}$$

Une estimation $\bar{g}_{ij,n}$ de $E\{g_{ij,n}\}$ est obtenue par filtrage passe bas de $g_{ij,n}$. Utilisons un indice t qui est incrémenté chaque fois qu'un nouvel exemple est présenté:

$$\bar{g}_{ij,n}(t) = (1-\beta) g_{ij,n}(t) + \beta \bar{g}_{ij,n}(t-1)$$

En posant $\delta_j = \frac{\partial e_Q}{\partial X_j}$, le gradient instantané de l'erreur par rapport à un poids s'écrit:

$$\begin{aligned} g_{ij,n} &= \frac{\partial e_Q}{\partial W_{ij,n}} \\ &= \frac{\partial e_Q}{\partial X_j}\frac{\partial X_j}{\partial W_{ij,n}} \\ &= \delta_j O_{i,n} \end{aligned}$$

Il nous reste à trouver une méthode pour calculer ce terme δ_j. En utilisant les règles élémentaires du calcul différentiel, on montre que δ_j s'exprime en fonction des δ_k relatifs aux neurones k de la couche suivante:

$$\begin{aligned} \delta_j &= \frac{\partial e_Q}{\partial X_j} \\ &= \frac{\partial e_Q}{\partial x_j}\frac{\partial x_j}{\partial X_j} \\ &= \left(\sum_{n=n_1}^{n_2} \frac{\partial e_Q}{\partial O_{j,n}}\frac{\partial O_{j,n}}{\partial x_j}\right) F'(X_j) \\ &= \left(\sum_{n=n_1}^{n_2} \left(\sum_{k=succ(j)} \frac{\partial e_Q}{\partial X_k}\frac{\partial X_k}{\partial O_{j,n}}\right)\frac{\partial O_{j,n}}{\partial x_j}\right) F'(X_j) \\ &= \left(\sum_{n=n_1}^{n_2} \left(\sum_{k=succ(j)} \delta_k W_{jk,n}\right) h'(x_j - n)\right) F'(X_j) \end{aligned}$$

Pour le cas particulier de la dernière couche, δ_j s'exprime aisément en fonction du type d'erreur choisi. Pour l'erreur quadratique, on a:

$$\begin{aligned} \delta_j &= \frac{\partial e_Q}{\partial x_j}\frac{\partial x_j}{\partial X_j} \\ &= (x_j - t_j)F'(X_j) \end{aligned}$$

1.6 Application à la réduction de dimensionnalité

Dans de nombreuses applications de traitement du signal, l'observation est un vecteur appartenant à un espace $\mathcal{E}$ de dimension N. Très souvent, les composantes de ce vecteur ne sont pas indépendantes, et sont en réalité restreintes à une hyper-surface $\mathcal{S}$ de dimension M inférieure à N.

La méthode d'analyse en composantes principales (ACP) permet de trouver la meilleure approximation de $\mathcal{S}$ par un hyperplan (meilleure approximation au sens des moindres carrés).

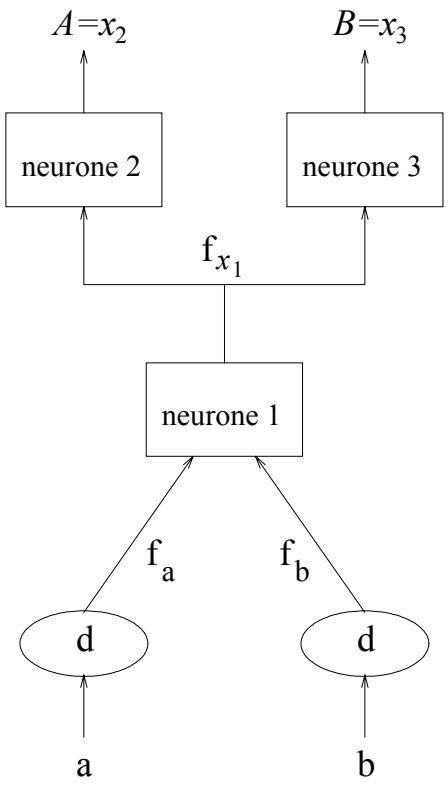


Figure 4: Réseau de neurones pour réduction de dimensionnalité

Lorsque l'hyper-surface $\mathcal{S}$ est fortement courbée, une approximation par un hyperplan n'est pas satisfaisante, et il n'existe actuellement aucune méthode pour traiter ce cas.

Notre objectif dans cette section est de montrer que notre modèle de réseau à Représentation Scalaire Distribuée est capable de traiter un tel problème, sans nécessiter de connaissance a priori sur la forme de $\mathcal{S}$. Afin de faciliter la visualisation des résultats, nous décrivons des expérimentations menées en dimension 2, mais notre modèle est bien entendu capable de traiter des dimensions supérieures.

Les données traitées sont des données synthétiques: il s'agit de vecteurs $\vec{v} = [a, b]^T$, aléatoirement choisis sur un quart de cercle (de rayon 1), selon une loi uniforme en fonction de l'angle. Ces données ont d'abord été traitées avec un réseau de neurones classique en "identity mapping" (2+1+2 neurones), qui fournit la même solution que l'analyse en composantes principales, à savoir la droite approximant au mieux le quart de cercle (au sens des moindres carrés).

Nous avons ensuite mis en œuvre notre modèle RSD (fig 4), avec une fonction

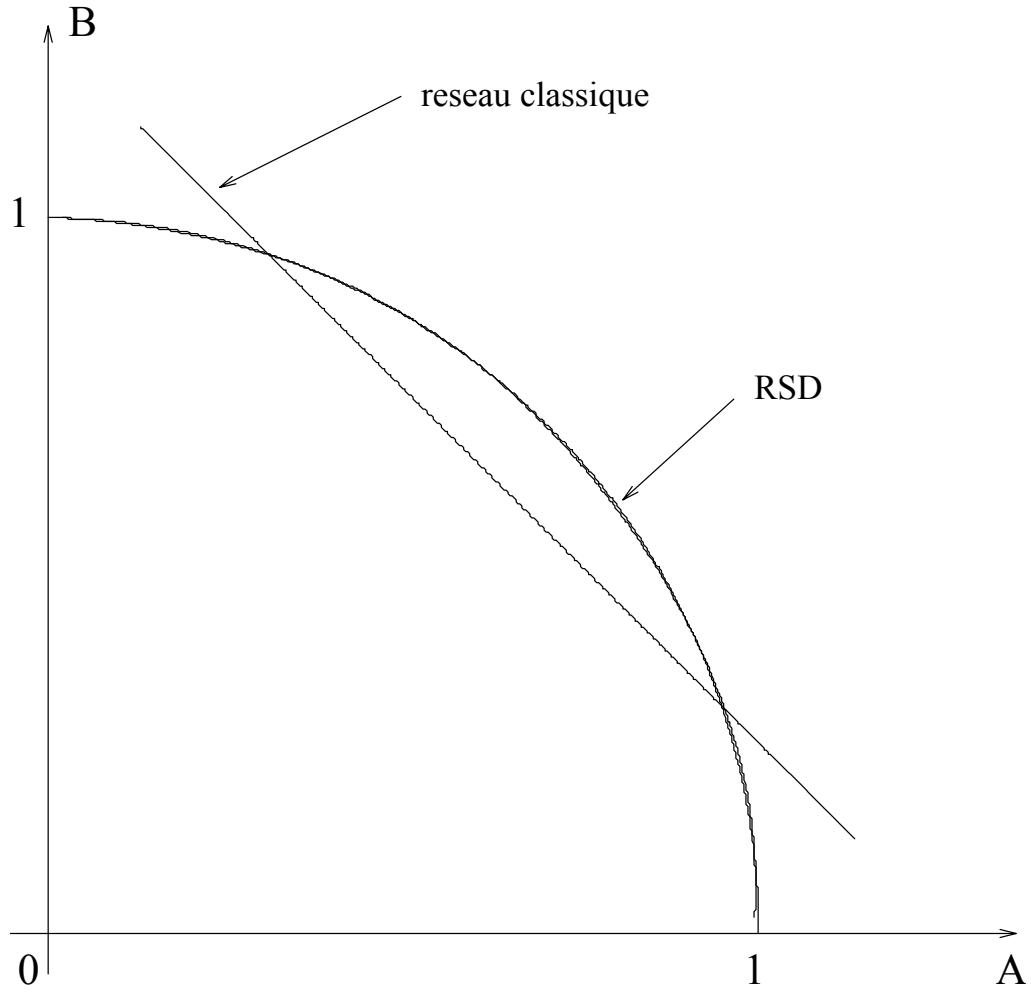


Figure 5: Réduction de dimensionnalité

génératrice de la RSD gaussienne. L'erreur quadratique est:

$$e_Q = \frac{1}{2}\left\{(A-a)^2 + (B-a)^2\right\}$$

Le tracé des coordonnées de sortie (A,B) fournit une excellente approximation du cercle (figure 5). Comme dans ce réseau, toute l'information a dû passer par x_1, on a bien réalisé une réduction de dimensionnalité.

Le tableau suivant indique les résultats obtenus en fin d'apprentissage (on a noté entre parenthèses le nombre de neurones par couche, hormis les seuils):

Réseau	e_{QM}
classique (2+1+2)	0.00389
RSD (2+1+2)	0.00002

Pour le réseau RSD mis en œuvre, on a:

$$h(u) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{u^2}{2\sigma^2}}$$

et

$$\begin{aligned} F(u) &= \tfrac{7}{2}(th(u)+1) \\ \sigma &= 1 \\ n_1 &= 0 \\ n_2 &= 7 \end{aligned}$$

Expérimentalement, on constate une mauvaise convergence de l'apprentissage lorsque l'écart type σ est choisi trop faible (inférieur à 0.5). Des problèmes de minima locaux ont été constatés pour ces valeurs. Le choix de σ est donc un compromis:

- Si σ est trop fort (supérieur à 3), les poids sont fortement lissés (voir expression de C_{ij}), et on restreint les capacités du réseau.
- Si σ est trop faible (inférieur à 0.5), le risque de blocage dans un minimum local devient non-négligeable.

Une solution pour éviter les 2 problèmes à la fois est de réduire σ en cours d'apprentissage. Pour les expérimentations, nous avons préféré choisir une valeur intermédiaire de l'écart type ($\sigma = 1$). Cette valeur permet d'obtenir un très bon résultat final, sans créer de problèmes de convergence de l'apprentissage.

Signalons également que la convergence est légèrement moins bonne lorsque la gaussienne est remplacée par un triangle.

1.7 Application à la prédiction

De nombreuses publications ont vanté les performances du perceptron multicouches lorsqu'il est utilisé comme classifieur. Par contre, ses performances en tant que prédicteur ont souvent été jugées médiocres. Nous présenterons au chapitre 9

(compression d'images) des résultats expérimentaux relatifs à la mise en œuvre de ce modèle comme prédicteur. Un gain important sur l'erreur de prédiction est obtenu, par rapport au modèle classique. Les performances d'un réseau utilisant ce type de neurone peuvent être expliquées par le fait que les connexions se comportent alors comme des fonctions adaptatives, et non plus comme de simples gains adaptatifs.

2 Nouveaux résultats relatifs à l'algorithme de Kohonen

[Nous utilisons les notations du chapitre 2, §3]

2.1 Introduction

L'algorithme d'auto-organisation de Kohonen présente d'intéressantes propriétés topologiques. Cependant, peu de résultats théoriques le concernant on été présentés, d'où un emploi relativement limité dans le domaine du traitement du signal. Dans la suite, nous présentons de nouveaux résultats théoriques, et des algorithmes voisins, plus proches des algorithmes classiquement utilisés en traitement du signal. L'attention est portée sur l'interprétation comme algorithme de minimisation d'une fonction d'énergie.

Après un rapide rappel de l'agorithme de Kohonen, une étude de son état d'équilibre nous permettra de proposer un algorithme similaire: l'algorithme VQN (Vector Quantization with Neighbourhood), qui ne nécessite pas le réglage d'une vitesse d'apprentissage. Nous montrerons que cet algorithme et l'algorithme de Kohonen minimisent, sous une condition d'organisation suffisante, une fonction d'énergie que nous nommerons e_{QMV} (Erreur Quadratique Moyenne avec Voisinage). Nous proposerons ensuite une légère modification de ces algorithmes afin de lever la condition. Ceci nous conduira aux algorithmes VQNf (Filtered VQN algorithm) et KHf (Filtered Kohonen algorithm). Enfin, des résultats expérimentaux en dimension 2 sont présentés. D'autres résultats sur une application réelle (compression d'images) sont disponibles dans le chapitre 9.

2.2 Rappel de l'algorithme de Kohonen

Afin d'homogénéiser les notations avec les algorithmes qui seront proposés par la suite, on présente l'algorithme de Kohonen dans le cas où l'indice d'itération (t) n'est incrémenté qu'à chaque présentation de tous les exemples. Les exemples sont extraits d'un ensemble d'apprentissage $\mathcal{A}$. On note T le nombre d'itérations, K le nombre d'entrées, et M le nombre de neurones:

1. $t = 0$
   Initialiser les vecteurs poids $\vec{W}_j = [W_{1j} \ ... \ W_{Kj}]^T$
2. $n = 1$
   Tirer une permutation aléatoire ρ de l'ensemble $\{1, 2, ..., card(\mathcal{A})\}$.
3. Présenter le vecteur $\vec{x}(\rho(n))$ à l'entrée du réseau
4. Calculer les sorties O_j des neurones
5. Déterminer le neurone k dont la sortie O_k est la plus faible
6. Modifier les poids conformément à:
$$\Delta\vec{W}_j = \alpha_{jk}(t).[\vec{x} - \vec{W}_j] \tag{1}$$
7. $n = n + 1$
   Si $n \leq card(A)$, aller en (3)
8. $t = t + 1$
   Si $t < T$, aller en (2)

Pour les réseaux de dimension 1, les termes $\alpha_{jk}(t)$ sont de la forme $\alpha(t, |j-k|)$. Nous proposons de prendre:
$$\alpha_{jk}(t) = \alpha_0 e^{-\frac{(j-k)^2}{2\sigma_t^2}} \tag{2}$$
L'écart type σ_t décroît avec t suivant une loi exponentielle:
$$\sigma_t = \sigma_0 \left(\frac{\sigma_{T-1}}{\sigma_0}\right)^{\frac{t}{T-1}}$$

Pour la clarté de la présentation, seuls les réseaux de dimension 1 sont considérés dans la suite. La généralisation aux dimensions supérieures est aisée car la dimension du réseau est entièrement déterminée par les valeurs des coefficients α_{jk}. Il suffit donc de remplacer $|j-k|$ par $d(j,k)$ dans l'équation 2, où d est la distance entre les

neurones j et k.

On prendra garde dans la suite à ne pas confondre la dimension du réseau (déterminée par les α_{jk}) et la dimension des entrées (K), de même que la distance entre les neurones ($|j-k|$ en 1D) et la distance entre les poids des ces neurones ($\|\vec{W}_j - \vec{W}_k\|$).

2.3 Etat de quasi-équilibre

Si les α_{jk} ne varient pas avec le temps, le réseau va se stabiliser dans un état tel que:

$$\forall j \qquad E_{\vec{x}\in\mathcal{A}}\{\Delta\vec{W}_j\} = \vec{0} \tag{3}$$

En pratique, les α_{jk} varient très lentement, et on peut supposer que le réseau est en permanence proche de cet état. Nous dirons que le réseau est en état de "quasi-équilibre". Nous nommerons à présent D_j l'ensemble suivant:

$$D_j = \{\vec{x} \in \mathcal{A} \mid \forall l \neq j \;\; \|\vec{x} - \vec{W}_j\|^2 < \|\vec{x} - \vec{W}_l\|^2\}$$

Et nous désignerons par $k(\vec{x})$ l'indice du vecteur $\vec{W}_j$ le plus proche de $\vec{x}$. En combinant les équations 1 et 3, on obtient:

$$\vec{W}_j = \frac{1}{E_{\vec{x}\in\mathcal{A}}\{\alpha_{jk(\vec{x})}\}} E_{\vec{x}\in\mathcal{A}}\{\alpha_{jk(\vec{x})}\vec{x}\}$$

En notant p_k la probabilité pour qu'un vecteur $\vec{x}$ appartienne au domaine D_k, et $\vec{b}_k$ le barycentre de D_k, on a:

$$E_{\vec{x}\in\mathcal{A}}\{\alpha_{jk(\vec{x})}\} = \sum_{k=1}^{M} \alpha_{jk} p_k$$

$$\begin{aligned} E_{\vec{x}\in\mathcal{A}}\{\alpha_{jk(\vec{x})}\vec{x}\} &= \sum_{k=1}^{M} E_{\vec{x}\in D_k}\{\alpha_{jk}\vec{x}\}.p_k \\ &= \sum_{k=1}^{M} \alpha_{jk} p_k \vec{b}_k \end{aligned}$$

D'où

$$\vec{W}_j = \frac{\sum_{k=1}^{M} \alpha_{jk} p_k \vec{b}_k}{\sum_{k=1}^{M} \alpha_{jk} p_k} \tag{4}$$

2.4 Quantification Vectorielle avec notion de voisinage

2.4.1 Idée générale

L'algorithme de Kohonen présente l'intérêt de conserver la topologie. Nous verrons au chapitre 9 comment cette propriété peut être exploitée à des fins de compression d'images.

Un problème lié à cet algorithme est le choix de la vitesse d'apprentissage. Pour contourner ce problème, nous proposons un nouvel algorithme d'apprentissage qui ne nécessite pas de paramètre "vitesse d'apprentissage". L'idée consiste à s'inspirer de l'algorithme des "k-means", et d'y introduire un voisinage entre les classes. On notera $\mathcal{A}$ l'ensemble d'apprentissage, et l'erreur quadratique moyenne est estimée par:

$$e_{QM} \simeq \frac{1}{card(\mathcal{A})} \sum_{\vec{x} \in \mathcal{A}} \|\vec{x} - \hat{\vec{x}}\|^2$$

L'algorithme proposé possède le même état de quasi-équilibre que l'algorithme de Kohonen, et peut donc être considéré comme similaire. En particulier, il présente les mêmes propriétés de conservation de la topologie.

2.4.2 Algorithme proposé (VQN)

L'algorithme proposé est le suivant (on note t l'indice d'itération):

1. $t = 0$
   Initialiser les prototypes $\{\vec{W}_1, \vec{W}_2, ..., \vec{W}_M\}$

2. Mettre à jour les domaines D_k:

$$D_k = \left\{\vec{x} \in \mathcal{A} \; tels \; que \; \forall j \neq k, \|\vec{x} - \vec{W}_k\|^2 < \|\vec{x} - \vec{W}_j\|^2\right\}$$

3. Calculer $e_{QM}(t)$ et arrêter si $\frac{e_{QM}(t) - e_{QM}(t-1)}{e_{QM}(t-1)} < \epsilon$, où ϵ est un réel positif.

4. Mettre à jour les poids:

$$\vec{W}_j = \frac{\sum_{k=1}^{M} \alpha_{jk} p_k \vec{b}_k}{\sum_{k=1}^{M} \alpha_{jk} p_k}$$

5. $t = t + 1$
   Aller en (2)

En pratique, on préfèrera fixer un nombre d'itérations T en fonction du temps de calcul disponible. L'étape (3) n'est donc généralement pas présente.

On rappelle que $\alpha_{jk}(t) = \alpha_0 e^{-\frac{(j-k)^2}{2\sigma_t}}$. En conséquence, le choix de la vitesse d'apprentissage α_0 est sans importance car il intervient aussi bien au numérateur qu'au dénominateur dans l'équation de mise à jour des poids.

Si les α_{jk} ne varient pas trop vite, on peut admettre que l'algorithme est en permanence en état de quasi-équilibre, c'est à dire que les classes et les prototypes sont simultanément à jour. Dans ce cas, notre algorithme est très proche de l'algorithme de Kohonen.

2.5 Quelques résultats théoriques

Propriété 1: Propriété du barycentre:

$$E_{\vec{x}\in D_k}\left\{\|\vec{x} - \vec{W}_l\|^2\right\} = \|\vec{W}_l - \vec{b}_k\|^2 + E_{\vec{x}\in D_k}\left\{\|\vec{x} - \vec{b}_k\|^2\right\}$$

Démonstration:

$$\begin{aligned}
E_{\vec{x}\in D_k}\left\{\|\vec{x} - \vec{W}_l\|^2\right\} &= E_{\vec{x}\in D_k}\left\{\|(\vec{x} - \vec{b}_k) + (\vec{b}_k - \vec{W}_l)\|^2\right\} \\
&= E_{\vec{x}\in D_k}\left\{\|\vec{x} - \vec{b}_k\|^2\right\} + \|\vec{W}_l - \vec{b}_k\|^2 + 2E_{\vec{x}\in D_k}\left\{(\vec{x} - \vec{b}_k).(\vec{b}_k - \vec{W}_l)\right\} \\
&= E_{\vec{x}\in D_k}\left\{\|\vec{x} - \vec{b}_k\|^2\right\} + \|\vec{W}_l - \vec{b}_k\|^2 \\
&\quad +2\left(E_{\vec{x}\in D_k}\{\vec{x}\}.\vec{b}_k - E_{\vec{x}\in D_k}\{\vec{x}\}.\vec{W}_l - \vec{b}_k.\vec{b}_k + \vec{b}_k.\vec{W}_l\right) \\
&= E_{\vec{x}\in D_k}\left\{\|\vec{x} - \vec{b}_k\|^2\right\} + \|\vec{W}_l - \vec{b}_k\|^2
\end{aligned}$$

Propriété 2:
Lorsque l'organisation des neurones est correcte, l'algorithme de Kohonen et l'algorithme de Quantification Vectorielle avec Voisinage (VQN) sont des algorithmes de minimisation de la fonction d'erreur suivante:

$$e_{QMV} = \sum_{l=1}^{M}\sum_{k=1}^{M} \alpha_{kl} p_k E_{\vec{x}\in D_k}\left\{\|\vec{x} - \vec{W}_l\|^2\right\}$$

Démonstration:

Lorsque les classes D_k sont fixées, minimiser e_{QMV} revient à minimiser:

$$e'_{QMV} = \sum_{l=1}^{M}\sum_{k=1}^{M} \alpha_{kl} p_k \left\{\|\vec{W}_l - \vec{b}_k\|^2\right\}$$

car d'après la propriété 1:

$$E_{\vec{x}\in D_k}\left\{\|\vec{x} - \vec{W}_l\|^2\right\} = \|\vec{W}_l - \vec{b}_k\|^2 + E_{\vec{x}\in D_k}\left\{\|\vec{x} - \vec{b}_k\|^2\right\}$$

Les dérivées partielles de e'_{QMV} par rapport aux poids s'écrivent:

$$\begin{aligned}\frac{\partial e'_{QMV}}{\partial W_{ij}} &= \frac{\partial}{\partial W_{ij}}\left\{\sum_{k=1}^{M} \alpha_{kj} p_k \|\vec{W}_j - \vec{b}_k\|^2\right\} \\ &= \frac{\partial}{\partial W_{ij}}\left\{\sum_{k=1}^{M} \alpha_{kj} p_k (W_{ij} - b_{ik})^2\right\} \\ &= 2\sum_{k=1}^{M} \alpha_{kj} p_k (W_{ij} - b_{ik})\end{aligned}$$

En annulant ces dérivées, on obtient:

$$\vec{W}_j = \frac{\sum_{k=1}^{M} \alpha_{kj} p_k \vec{b}_k}{\sum_{k=1}^{M} \alpha_{kj} p_k}$$

Ce qui correspond bien à l'étape 4 de l'algorithme VQN, car $\alpha_{jk} = \alpha_{kj}$.

Lorsque les poids sont fixés, e_{QMV} peut s'écrire:

$$\begin{aligned}e_{QMV} &= \sum_{l,k} \alpha_{kl} \frac{card\ D_k}{card\ \mathcal{A}} \frac{\sum_{\vec{x}\in D_k} \|\vec{x} - \vec{W}_l\|^2}{card\ D_k} \\ &= \frac{1}{card\ \mathcal{A}} \sum_{\vec{x}\in\mathcal{A}} \sum_{l=1}^{M} \alpha_{k(\vec{x}),l} \|\vec{x} - \vec{W}_l\|^2\end{aligned}$$

Pour chaque vecteur $\vec{x}$, on doit donc minimiser:

$$\sum_{l=1}^{M} \alpha_{k(\vec{x}),l} \|\vec{x} - \vec{W}_l\|^2$$

On a vu que, pour les réseaux mono-dimensionnels, $\alpha_{kl} = \alpha(|k-l|)$. Nous poserons à présent:

$$\begin{aligned} h_{k-l} &= \alpha_{kl} \\ f_l &= \|\vec{x} - \vec{W}_l\|^2 \end{aligned}$$

Le terme à minimiser s'écrit alors (en désignant par "$*$" le produit de convolution):

$$\begin{aligned} \sum_l \alpha_{kl} \|\vec{x} - \vec{W}_l\|^2 &= \sum_l h_{k-l} f_l \\ &= (h * f)_k \end{aligned}$$

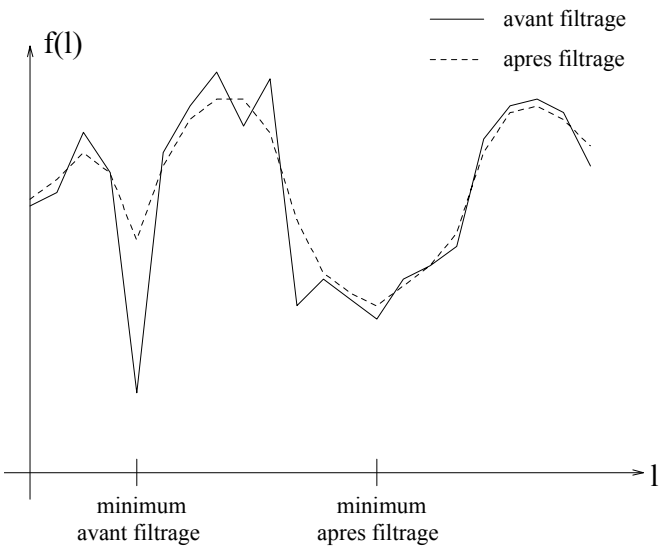


Figure 6: Filtrage (réseau mal organisé)

Pour minimiser e_{QMV}, on doit donc associer $\vec{x}$ au domaine D_k tel que k soit la position du minimum de f <u>après</u> filtrage par h. Avec la forme que nous avons proposée pour les α_{kl}, le filtre h est un filtre gaussien.

Mais l'étape 2 de l'algorithme VQN affecte $\vec{x}$ au domaine D_k tel que k est la position du minimum de f <u>avant</u> filtrage. L'algorithme VQN et l'algorithme de Kohonen ne peuvent donc rigoureusement être considérés comme des algorithmes de minimisation de e_{QMV} que lorsque ces 2 minima sont confondus. Ceci sera le cas si le réseau est bien organisé, c'est à dire si les poids de neurones voisins sont assez proches. Les figures 6 et 7 illustrent cet énoncé. Normalement, après quelques itérations de l'algorithme, les neurones commencent effectivement à s'organiser,

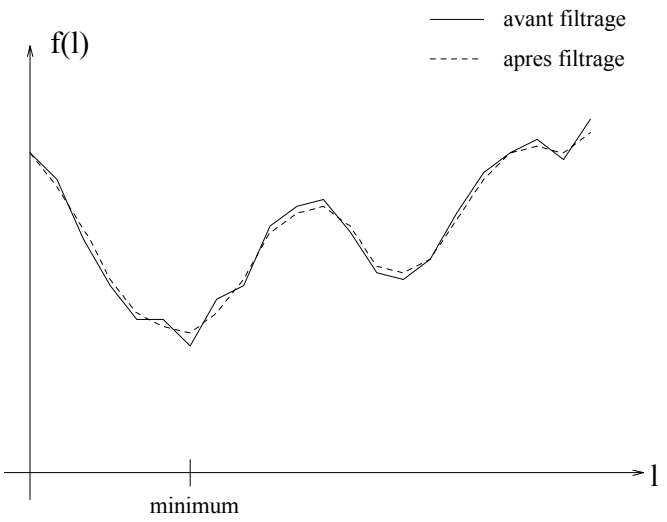


Figure 7: Filtrage (réseau bien organisé)

et l'on peut donc sans grand risque d'erreur interpréter ces algorithmes comme des algorithmes de minimisation de e_{QMV}. De plus, l'initialisation des poids est généralement effectuée de telle sorte que l'organisation soit effective dès le début, et cette propriété est conservée durant l'apprentissage.

2.6 Deux algorithmes de minimisation de e_{QMV}

L'objet de ce paragraphe est de proposer deux algorithmes qui minimisent rigoureusement la fonction d'erreur e_{QMV}.

Algorithme de Kohonen filtré (KHf)

1. $t = 0$
   Initialiser les vecteurs poids $\vec{W}_j = [W_{1j} \ ... \ W_{Kj}]^T$
2. $n = 1$
   Tirer une permutation aléatoire ρ de l'ensemble $\{1, 2, ..., card(\mathcal{A})\}$
3. Présenter le vecteur $\vec{x}(\rho(n))$ à l'entrée du réseau
4. Calculer les sorties O_j des neurones

5. Filtrer les sorties par $h_{k-l} = \alpha_{kl}(t)$
6. Déterminer le neurone k dont la sortie filtrée est la plus faible
7. Modifier les poids conformément à:

$$\Delta\vec{W}_j = \alpha_{jk}(t).[\vec{x} - \vec{W}_j]$$

8. $n = n + 1$
   Si $n \leq card(\mathcal{A})$, aller en (3)
9. $t = t + 1$
   Si $t < T$, aller en (2)

Algorithme VQN filtré (VQNf)

1. $t = 0$
   Initialiser les prototypes $\{\vec{W}_1, \vec{W}_2, ..., \vec{W}_M\}$

2. Mettre à jour les classes D_k:

$$D_k = \left\{ \vec{x} \in \mathcal{A} \; tels \; que \; \forall j \neq k, \sum_{l=1}^{M} \alpha_{kl} \|\vec{x} - \vec{W}_l\|^2 < \sum_{l=1}^{M} \alpha_{jl} \|\vec{x} - \vec{W}_l\|^2 \right\}$$

3. Calculer $e_{QMV}(t)$ et arrêter si $\frac{e_{QMV}(t) - e_{QMV}(t-1)}{e_{QMV}(t-1)} < \epsilon$, où ϵ est un réel positif.

4. Mettre à jour les poids:

$$\vec{W}_j = \frac{\sum_{k=1}^{M} \alpha_{jk} p_k \vec{b}_k}{\sum_{k=1}^{M} \alpha_{jk} p_k}$$

5. $t = t + 1$
   Aller en (2)

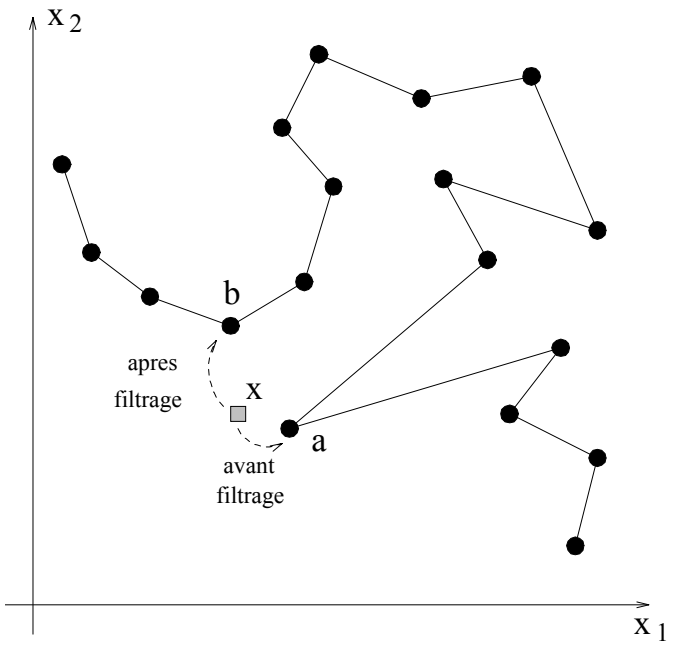


Figure 8: Influence du filtrage en 2D

En pratique, on préfèrera fixer un nombre d'itérations T en fonction du temps de calcul disponible. L'étape (3) n'est donc généralement pas présente.

La figure 8 illustre la différence, sur un cas où les entrées sont de dimension 2, entre les algorithmes ci-dessus et les algorithmes de Kohonen et VQN. Le vecteur $\vec{x}$ représenté sur cette figure sera associé au neurone (a) avec les algorithmes de Kohonen et VQN, car ce neurone est le plus proche de $\vec{x}$. Par contre, avec les algorithmes ci-dessus, il sera associé au neurone (b). Ce neurone n'est pas le plus proche de $\vec{x}$, mais il est soutenu par son voisinage qui forme autour de lui un bloc relativement proche de $\vec{x}$. En d'autres termes, la distance filtrée entre (b) et $\vec{x}$ est plus faible que la distance filtrée entre (a) et $\vec{x}$.

D'un point de vue intuitif, on pourrait penser que ces 2 derniers algorithmes vont converger plus vite que les algorithmes de Kohonen et VQN. En effet, les neurones trop éloignés de leurs voisins vont être défavorisés, et vont donc revenir rapidement à proximité de ces voisins. En conséquence, l'organisation du réseau serait plus rapide.

2.7 Forme approchée de e_{QMV}

Lorsque l'influence du voisinage n'est pas trop importante, les vecteurs $\vec{W}_k$ sont proches des barycentres de leurs classes respectives ($\vec{b}_k$). On a alors l'approximation suivante pour e_{QMV}:

$$\begin{aligned}
e_{QMV} &= \sum_{l=1}^{M}\sum_{k=1}^{M} \alpha_{kl} p_k E_{\vec{x}\in D_k}\{\|\vec{x}-\vec{W}_l\|^2\} \\
&= \sum_{l=1}^{M}\sum_{k=1}^{M} \alpha_{kl} p_k \left(\|\vec{W}_l-\vec{b}_k\|^2 + E_{\vec{x}\in D_k}\{\|\vec{x}-\vec{b}_k\|^2\}\right) \\
&\simeq \sum_{l=1}^{M}\sum_{\substack{k=1\\k\neq l}}^{M} \alpha_{kl} p_k \|\vec{W}_l-\vec{W}_k\|^2 + \sum_{l=1}^{M}\left(\alpha_{ll} p_l \|\vec{W}_l-\vec{b}_l\|^2 + \alpha_{ll} p_l E_{\vec{x}\in D_l}\{\|\vec{x}-\vec{b}_l\|^2\}\right) \\
&\quad + \sum_{l=1}^{M}\sum_{\substack{k=1\\k\neq l}}^{M} \alpha_{kl} p_k E_{\vec{x}\in D_k}\{\|\vec{x}-\vec{b}_k\|^2\} \\
&\simeq \sum_{l=1}^{M}\sum_{\substack{k=1\\k\neq l}}^{M} \alpha_{kl} p_k \|\vec{W}_l-\vec{W}_k\|^2 + \sum_{l=1}^{M} \alpha_{ll} p_l \|\vec{W}_l-\vec{b}_l\|^2 + \sum_{k=1}^{M}\left(\sum_{l=1}^{M}\alpha_{kl}\right) p_k E_{\vec{x}\in D_k}\{\|\vec{x}-\vec{b}_k\|^2\}
\end{aligned}$$

Cette formule approchée fait apparaître un terme dépendant des distances entre les prototypes. Les propriétés de conservation de la topologie des algorithmes précédents apparaissent donc explicitement.

2.8 Quelques résultats expérimentaux en dimension 2

Nous présentons ci-dessous des résultats expérimentaux en dimension 2, ce qui présente l'avantage de permettre une visualisation aisée des résultats. D'autres résultats sont disponibles dans le chapitre 9 (compression d'images).

L'ensemble d'apprentissage $\mathcal{A}$ contient 4092 vecteurs de dimension $K = 2$. Il s'agit de blocs de 2x1 pixels extraits d'une image (les composantes d'un vecteur sont donc les luminances l_1 et l_2 des 2 pixels du bloc). L'image est à 256 niveaux de gris. La figure 9 représente la racine cubique de la densité de probabilité conjointe $p(l_1, l_2)$. Une image contient beaucoup de zones uniformes, d'où une forte densité sur la diagonale ($l_1 = l_2$).

Les expérimentations ont été menées avec $M = 32$ prototypes. Le nombre d'itérations accordées est $T = 100$. Pour les algorithmes présentant une notion de voisinage, on a $\sigma_0 = 4.0$ et $\sigma_{T-1} = 0.2$. L'initialisation des prototypes est réalisée

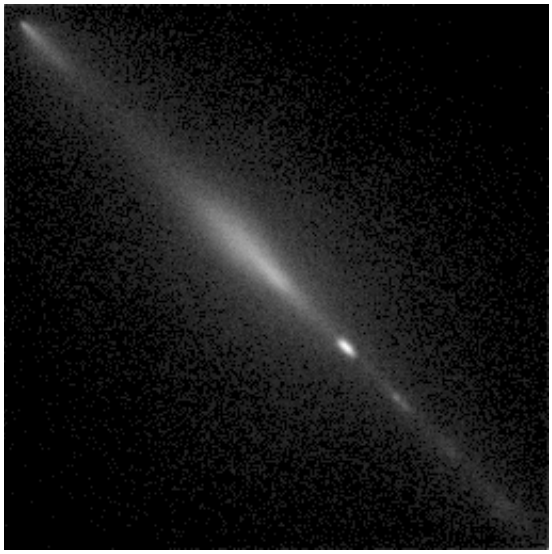


Figure 9: Racine cubique de la densité de probabilité pour des blocs 2x1

comme suit (sauf pour l'algorithme LBG qui possède sa propre initialisation):

$$\forall i,j \qquad W_{ij} = \frac{255(j-1)}{M-1} + b_{ij}$$

b_{ij} est un bruit de densité uniforme entre -5 et $+5$.

Les figures 10 et 11 permettent de comparer les courbes d'apprentissage de divers algorithmes. On représente l'évolution de l'erreur quadratique moyenne (et non de e_{QMV}, car cela n'aurait pas de sens pour les algorithmes classiques).

La figure 10 permet de comparer les courbes d'apprentissage de l'algorithme de Kohonen (pour $\alpha_0 = 0.02$ et $\alpha_0 = 0.10$) et de l'algorithme VQN. Une assez forte vitesse d'apprentissage ($\alpha_0 = 0.10$) conduit à un apprentissage relativement bruité et à un résultat final moins bon. La valeur $\alpha_0 = 0.02$, qui correspond au meilleur choix que nous ayons pu obtenir, conduit à une courbe d'apprentissage très proche de VQN, ce qui confirme la similarité des deux algorithmes. L'intérêt de VQN est qu'il ne nécessite pas le réglage de la vitesse d'apprentissage.

La figure 11 permet de comparer les courbes d'apprentissage des algorithmes VQN, VQNf, "k-means", et LBG. L'état initial est le même pour tous les algorithmes (et correspond à $e_{QM} = 161.8$), sauf pour LBG qui a effectué au préalable 80 itérations avec des valeurs croissantes de M (cf chap 2), afin de créer son état initial.

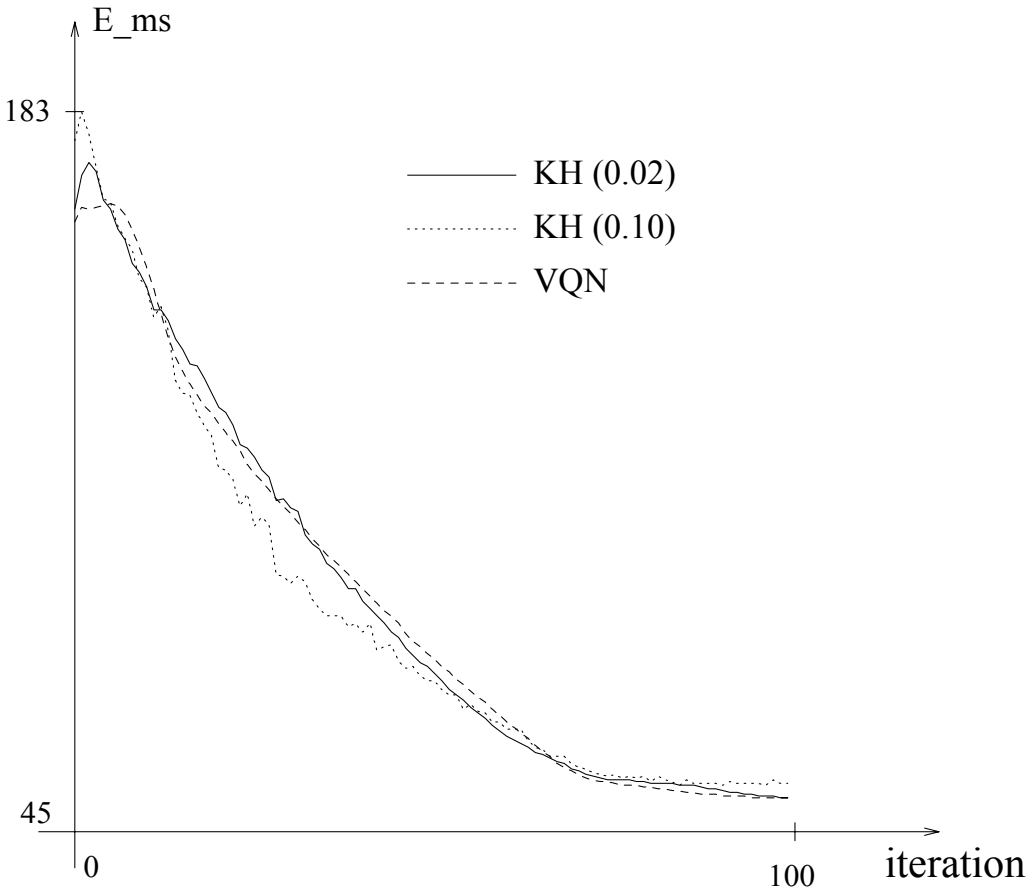


Figure 10: Comparaison des courbes d'apprentissage: Kohonen et VQN

L'observation de ces courbes montre que les algorithmes classiques convergent plus vite que VQN et VQNf (qui sont limités par la vitesse de décroissance du voisinage). Par contre, les algorithmes topologiques conduisent à une meilleure solution en fin d'apprentissage. Les intéractions topologiques ont donc pour effet de réduire le risque de piégeage dans un minimum local de l'erreur quadratique.

En dimension 2, il est possible de représenter les vecteurs $\vec{W}_j$ par des points du plan. Sur la figure 12, on a représenté l'état de ces vecteurs en fin d'apprentissage pour les différents algorithmes. Des segments relient $\vec{W}_j$ à $\vec{W}_{j+1}$ afin d'indiquer la topologie. L'état initial était relativement organisé, et l'algorithme des "k-means" a partiellement conservé cette organisation, bien que cet algorithme ne tienne pas compte du voisinage. Les algorithmes KH, VQN et VQNf présentent une bonne organisation topologique (sauf pour KH lorsque la vitesse d'apprentissage est trop élevée).

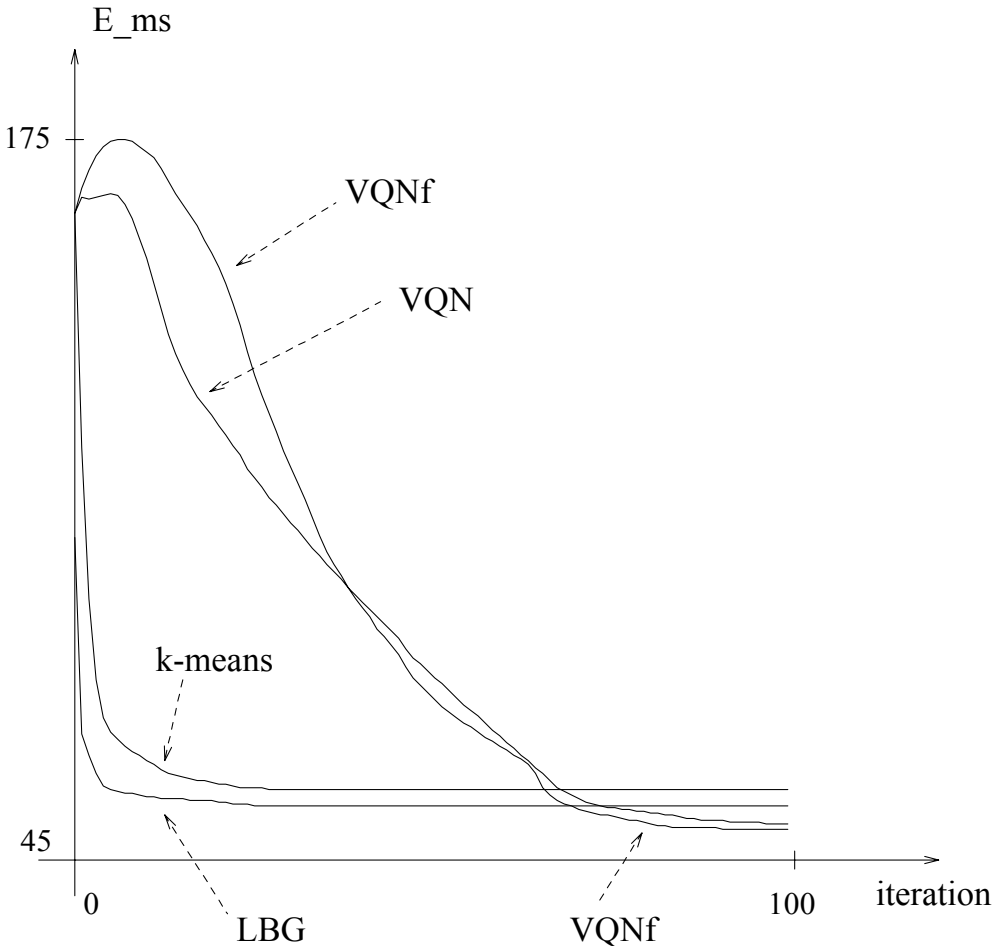


Figure 11: Comparaison des courbes d'apprentissage: k-means, LBG, VQN et VQNf

Le tableau ci-dessous indique les valeurs finales de e_{QM} pour ces différents algorithmes:

algorithme	e_{QM}
VQNf	50.60
VQN	51.59
KH(0.02)	51.61
KH(0.10)	54.38
LBG	54.84
k-means	57.57

Il est intéressant de voir dans quelle mesure ces algorithmes sont capables de se réorganiser lorsque l'état initial est totalement désordonné. Pour cela, on a initialisé

les vecteurs $\vec{W}_{ij}$ comme suit:

$$\forall i, j \qquad W_{ij} = b'_{ij}$$

où b'_{ij} est un bruit de densité uniforme entre 0 et 255. La figure 13 indique l'état initial, ainsi que les états atteints par les différents algorithmes. LBG n'est pas représenté car il possède sa propre initialisation, et conduit donc au même résultat que sur la figure 12.

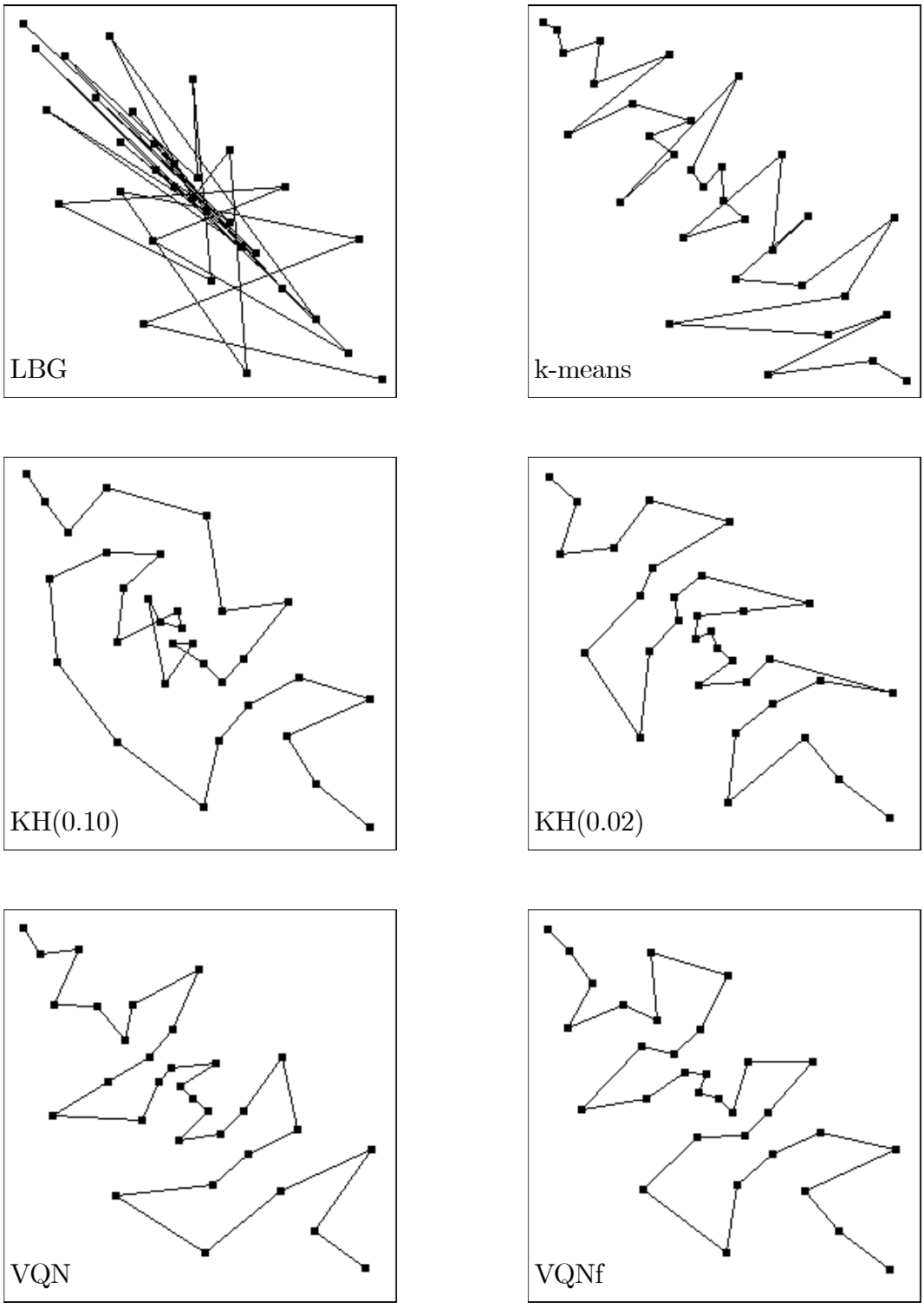


Figure 12: Etat des prototypes en fin d'apprentissage (initialisation diagonale)

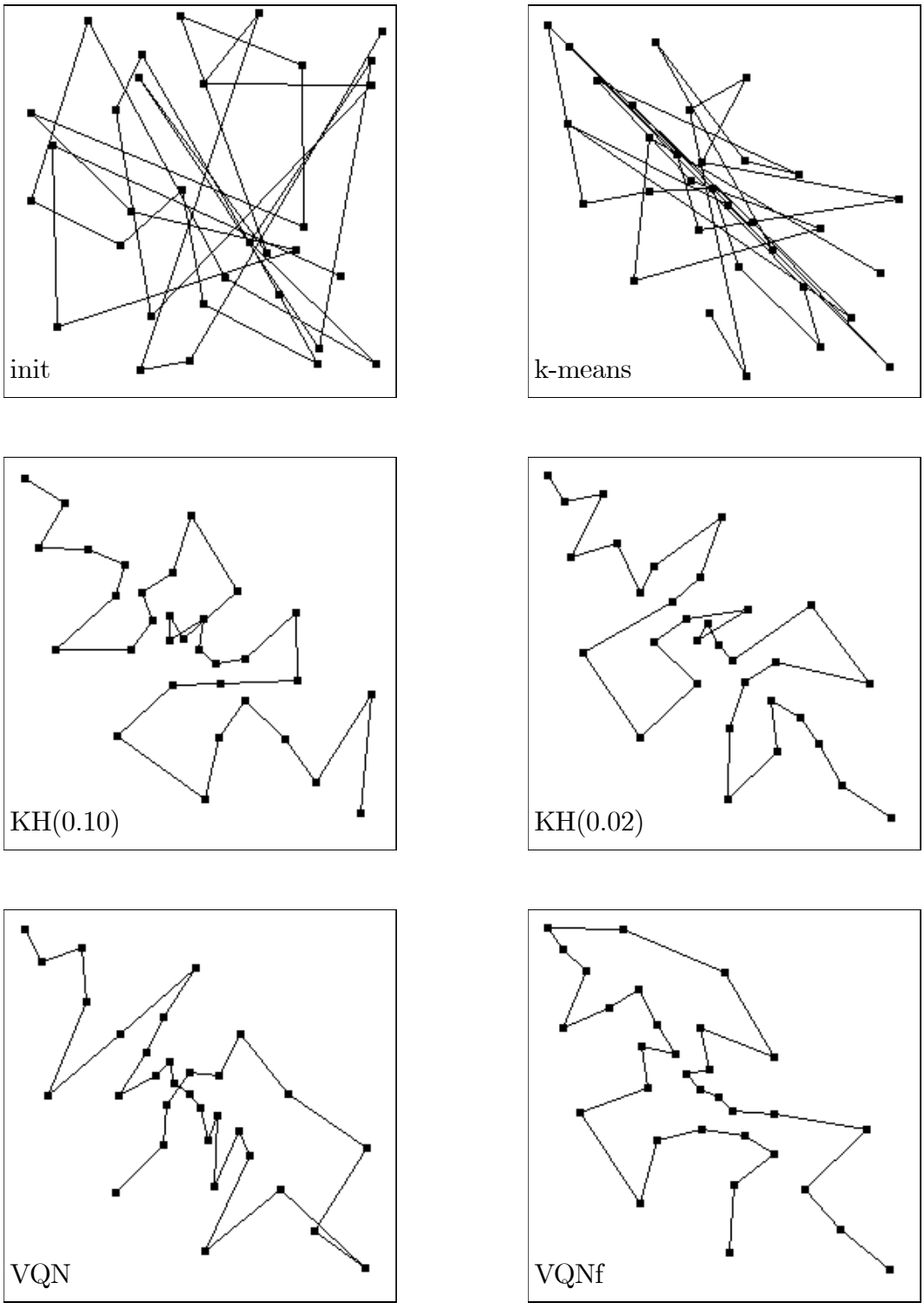


Figure 13: Etat des prototypes en fin d'apprentissage (initialisation aléatoire)

Chapitre 5:

DISCRIMINATION DE TEXTURES

La texture est une caractéristique importante pour la segmentation de divers types d'images, des images aériennes aux images médicales. Le système visuel humain est extrèmement performant dans ce domaine. Ainsi, sur une scène naturelle en extérieur on distingue sans difficulté les différentes textures: l'herbe, le feuillage, le sable, etc (fig 11). Cependant, la segmentation automatique d'images par analyse de texture est toujours l'un des problèmes les plus difficiles dans le domaine du traitement d'images; problème qui reste d'ailleurs ouvert car il n'y a pas de définition universellement acceptée de la notion de texture. Intuitivement, la notion de texture est liée à l'aspect homogène d'une surface. Une propriété essentielle de la perception texturelle est son invariance par translation: une texture laisse la même impression au système visuel, quelque soit la partie de la texture qui est observée.

Un bref rappel des techniques les plus couramment employées pour la classification de textures nous permettra de dégager un point commun à ces techniques: l'extraction d'un petit nombre de paramètres caractéristiques (quelques unités à quelques dizaines). Ceci est principalement dû au fait que la plupart des classifieurs disponibles nécessitent un bon choix de paramètres (de préférence non redondants, hautement discriminants, et tels qu'une distance euclidienne dans l'espace paramétrique soit significative). Le danger inhérent à ce type d'approche est le risque de recouvrement de classes dans l'espace paramétrique lorsque la base de données devient importante (ce qui est le cas pour les applications opérationnelles).

Nous proposons ici une approche différente qui n'est pas sujette à ce problème, car les paramètres extraits sont suffisants pour caractériser totalement la texture (i.e. la connaissance de ces paramètres suffit pour synthétiser une texture tout à fait semblable à l'originale). Ces paramètres sont les échantillons de l'histogramme et de l'autocovariance de la texture, et sont au nombre de quelques centaines. Nous montrerons qu'un perceptron multicouches se révèle suffisamment performant pour apprendre à reconnaitre les fonctions d'autocovariance et les histogrammes des différentes textures à classifier. Une comparaison avec d'autres classifieurs, alimentés par les mêmes paramètres, montrera l'avantage du perceptron multicouches. Enfin, nous validerons notre approche sur diverses images des domaines visible et Infra-Rouge.

1 Méthodes classiques

Nous nous situons ici dans le cadre de la classification supervisée de textures, c'est à dire dans le cas où l'on a prédéfini les classes (par exemple: ciel, terre, habitations, ...), et où l'on dispose a priori d'échantillons représentatifs de ces textures. Il existe également des méthodes de classification non-supervisée, qui segmentent une image en zones, le regroupement des pixels de l'image dans une même zone se faisant sur un critère de proximité dans l'espace paramétrique. Le résultat obtenu (segmentation de l'image) dépend donc uniquement du choix des paramètres. Les méthodes non-supervisées peuvent être utiles pour découper une image en zones, et servir de prétraitement à des étages supérieurs (reconnaissance d'objets, ...). Mais les zones ne correspondent pas obligatoirement à une entité physique (ciel, feuillage, ...).

Un grand nombre de méthodes a été proposé pour la discrimination de textures. Toutefois, il est possible de les regrouper en deux grandes familles:

1. Les méthodes statistiques
2. Les méthodes structurelles

De bonnes revues sont fournies par [*Haralick*79], [*Haralick*73], et [*Wechsler*80]. Les applications sont très variées, et couvrent divers types d'images, des images médicales aux images aériennes [*Lumia*83] [*Holmes*84]. Comme Kashyap [*Kashyap*86] l'a très justement noté, le point fondamental dans toutes ces méthodes est "le choix d'un ensemble de caractéristiques qui permettent de réduire la dimension des données à une quantité acceptable pour le calculateur, tout en préservant la plus grande partie de l'information discriminante".

Les méthodes statistiques considèrent la texture comme un champ aléatoire à deux dimensions, et les caractéristiques requises sont obtenues par des approches telles que les matrices de co-occurence [*Haralick*73] [*Davis*79] [*Sun*83] [*Vickers*82], la fréquence spatiale [*Jernigan*84] [*Coggins*85] [*Hsiao*89] [*Clark*87] [*Bovick*90] [*Reed*90], la corrélation [*Chen*83] [*Faugeras*80], ou les modèles paramétriques [*Wechsler*79] [*Cross*83] [*Kashyap*86] [*Derin*86] [*Derin*87]. Quelques études comparatives non-exhaustives ont été proposées [*Conners*80] [*Ronsin*85]. Elles montrent que les méthodes basées sur les matrices de co-occurence sont généralement les plus performantes, bien que des méthodes moins gourmandes en puissance de calcul peuvent donner des résultats comparables sur des ensembles réduits de textures [*Ronsin*85].

Les méthodes structurelles caractérisent la texture par des primitives élémentaires appelés "texels", et par l'arrangement spatial de ces primitives [*Ahuja*81] [*Tomita*82]

[*Leu*85] [*Vilnrotter*86] [*Eichmann*88] [*Tüceryan*90]. Bien que probablement plus proches de la façon dont le système visuel travaille, les méthodes structurelles sont généralement plus complexes que les méthodes statistiques, et réagissent assez mal en présence de textures faiblement structurées telles que l'herbe, le feuillage, la laine, et un grand nombre de textures naturelles.

On constate donc qu'aux deux grandes familles précédemment citées correspondent deux définitions de la texture: une définition statistique et une définition structurelle. Il semble qu'une définition statistique soit mieux adaptée à une approche neuronale. En effet, l'expérience montre que les réseaux de neurones présentent leurs pleines potentialités lorsqu'il s'agit de traiter des données floues, bruitées, partiellement incohérentes. Lorsque les données à traiter sont de haut niveau (ce qui sera le cas avec une approche structurelle), une approche système expert est généralement supérieure. De plus, un grand nombre de textures naturelles peuvent être vues comme des champs aléatoires.

2 Méthode proposée

2.1 Idées générales

En réalisant des expériences psychovisuelles, Gagalowicz [*Gagalowicz*83] a obtenu un résultat particulièrement intéressant concernant la discrimination visuelle de textures aléatoires: "Deux textures naturelles possédant les mêmes histogrammes locaux et les mêmes fonctions d'autocovariance ne sont pas visuellement discriminables". Cette conjecture a été vérifiée sur de nombreux exemples de textures naturelles.

Ce résultat limite l'ensemble des paramètres à prendre en compte par un système qui vise à réaliser la discrimination de textures "comme l'œil le fait". Cependant, la quantité de données à traiter est toujours énorme. Une première solution est d'extraire un faible nombre de paramètres de l'histogramme et de l'autocovariance (par exemple l'entropie de l'histogramme, etc). Mais le problème résultant est le risque de perte d'information significative.

La solution que nous proposons est d'utiliser un classifieur suffisamment puissant pour traiter directement l'histogramme et l'autocovariance. Un réseau de neurones multicouches semble être capable de réaliser une telle tâche. Cette approche diffère de la plupart des méthodes traditionnelles de par le fait qu'il n'y a pas de réduction draconienne des données (par exemple, les méthodes à base de matrices de

co-occurrence n'utilisent jamais directement ces matrices, mais simplement un petit nombre de paramètres ad-hoc extraits de ces matrices).

Durant une phase d'apprentissage préalable, ce système aura appris, à partir d'exemples, à caractériser diverses textures. Un réseau de neurones multicouches recevra en entrée l'autocovariance et l'histogramme de la texture (calculés sur une fenêtre d'observation), et fournira une indication de classe et de confiance (la classe et la confiance seront affectées au pixel central de la fenêtre d'observation). On met ainsi à la disposition du réseau toute l'information pertinente pour la discrimination des fonds, ce qui n'est pas le cas avec les systèmes classiques qui n'exploitent généralement qu'un nombre restreint de paramètres.

En contrepartie, l'apprentissage sera certainement plus long du fait de la complexité de la métrique pertinente sur l'espace paramétrique. En effet, la majorité des méthodes classiques exploitent des paramètres de texture qui sont tels que chaque paramètre (variance, énergie dans une bande de fréquence du spectre, etc) apporte une information significative indépendamment des autres. Une simple distance euclidienne est alors significative. Ceci n'est pas le cas dans le traitement que nous proposons car, par exemple, un point de la fonction d'autocovariance apporte une information quasi-nulle s'il est pris indépendamment des autres. Mais par contre, l'information globale apportée par l'autocovariance et de l'histogramme est très élevée car il est souvent possible de synthétiser une texture très semblable à l'originale à partir de cette donnée. Le paragraphe suivant a pour objectif de vérifier ce point.

2.2 Vérification de la pertinence des paramètres

Afin de vérifier la pertinence des paramètres choisis, nous combinons un système d'identification et un système de synthèse (fig 1). L'identification réside dans l'extraction de l'autocovariance et de l'histogramme d'un échantillon de texture naturelle. Puis on crée de la texture synthétique de telle sorte qu'elle ait même autocovariance et histogramme que la texture naturelle. Si la texture synthétisée a même apparence que la texture d'origine, cela signifie que l'ensemble {autocovariance + histogramme} est suffisant pour caractériser une texture.

Le vecteur paramètre que nous allons extraire est donc:

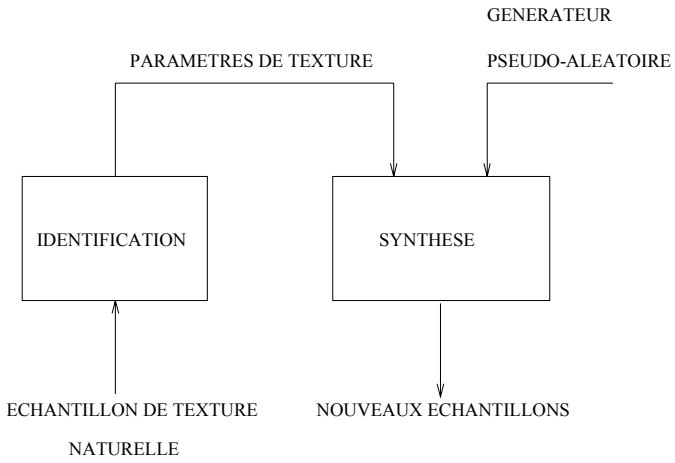


Figure 1: Synthèse de textures

$$\vec{B}=\left|\begin{array}{c}\vec{H}\\ \\ \vec{C}\end{array}\right| \quad \begin{array}{rl} \text{où } \vec{H} & = \text{vecteur histogramme local} \\ \vec{C} & = \text{vecteur moments centrés du second ordre (auto-covariance)}\end{array}$$

Nous ne décrirons pas en détail la méthode de synthèse (cf [*Gagalowicz*83] [*Henocq*90]), mais nous nous contenterons ci-dessous d'en dresser les grandes lignes. La méthode de synthèse est une méthode itérative, qui consiste à modifier progressivement l'échantillon de la nouvelle texture de sorte que ses paramètres se rapprochent de plus en plus de ceux de la texture naturelle. Notons $\hat{\vec{H}}$ et $\hat{\vec{C}}$ les paramètres de la nouvelle texture en cours de synthèse et

$$\hat{\vec{B}}=\left|\begin{array}{c}\hat{\vec{H}}\\ \\ \hat{\vec{C}}\end{array}\right| \quad \text{son vecteur paramètre.}$$

On se définit une erreur qui est:

$$\text{E}=\| \hat{\vec{C}} - \vec{C} \|^2 + \alpha \| \hat{\vec{H}} - \vec{H} \|^2$$

avec α suffisamment grand (typ. $5\text{x}10^8$).

En effet, il n'est pas souhaitable de minimiser $\| \hat{\vec{B}} - \vec{B} \|^2$ car $\vec{B}$ contient des grandeurs qui n'ont pas la même dimension.

Le principe de synthèse consiste d'abord à générer une image aléatoire dont l'histogramme est identique à celui de la texture naturelle (ceci peut être aisément réalisé en donnant aux pixels des luminances aléatoires suivant une distribution de probabilité correspondant à l'histogramme). Puis on réitère un grand nombre de fois l'opération qui consiste à tirer un pixel au hasard et à lui donner la luminance qui améliore le mieux l'erreur. Il serait bien entendu coûteux de recalculer l'autocovariance à chaque itération. Mais si l'on prend α assez grand, l'histogramme $\hat{\vec{H}}$ est contraint à rester très proche de $\vec{H}$, ce qui autorise des approximations conduisant à une expression récurrente de l'autocovariance en fonction de sa valeur à l'itération précédente.

Reste à savoir sur quelle extension on doit calculer l'autocovariance. Selon des expériences psycho-visuelles réalisées par Gagalowicz[*Gagalowicz*83], l'œil humain est peu sensible aux corrélations entre des points vus sous un angle solide supérieur à 9'. Or l'observateur humain est capable de classifier sans trop de problèmes les textures des images de notre base de données lorsque celles-ci sont présentées sur une station de travail. Sur ce type de station, la hauteur du pixel est d'environ 0.35mm, et l'observateur est à une distance de l'écran de l'ordre de 50cm. Dans ces conditions, un simple calcul montre que 9' d'angle correspondent à 4 pixels. Compte tenu du fait que la mesure donnée par Gagalowicz est très approximative, et que la distance écran-observateur peut varier énormément, on s'accorde une bonne marge en prenant une une extension de 15 pixels. Comme l'autocovariance est symétrique par rapport à l'origine, elle est calculée sur un demi-plan seulement.

La synthèse d'images de Brodatz [*Brodatz*66] a été réalisée en utilisant la méthode précédente (fig 5). Chaque texture fait 128x128 pixels, et l'identification a été réalisée, pour chaque texture, sur une fenêtre de 64x64 pixels. Notons que le temps de calcul requis est très important, malgré l'utilisation d'expressions récurrentes. Le temps CPU nécessaire sur SUN4 est d'environ 3.5 secondes par itération. Un total de 2x128x128 = 32768 itérations a été réalisé pour la synthèse de chaque texture (soit 32H CPU dans chaque cas). Les résultats obtenus montrent que les textures synthétisées sont très semblables aux textures naturelles (sauf pour les textures très structurées). Rappelons qu'il convient d'observer ces images à une distance suffisante car la texture est par définition une impression d'ensemble, et non pas le résultat d'une analyse du détail. Le fait que les résultats soient moins bons pour les textures très structurées est tout à fait naturel car nous avons adopté un modèle stochastique et non un modèle structurel de la texture.

Le même processus a été appliqué aux textures "ciel", "terre", et "habitations" définies sur images IR. L'identification a été réalisée, pour chaque texture, sur un bloc de 64x64 pixels des figures 6, 7, et 8. Les résultats de synthèse sont fournis sur la figure 9. Les images synthétiques "ciel" et "terre" ont même texture que

les images naturelles. Cette ressemblance est moins vraie pour la synthèse de la texture "habitations", qui est plus structurée. On retrouve toutefois les principaux aspects des zones d'habitation: alignements, alternance de zones sombres(toits) et claires(murs),...

En conclusion il n'y a donc pas de perte d'information significative pour la discrimination de texture dans le prétraitement que nous réalisons (du moins tant que les textures ne sont pas trop structurées). Remarquons que ceci n'est pas le cas avec les méthode habituelles, où le nombre de paramètres extraits est limité et serait tout à fait insuffisant pour resynthétiser des textures de même apparence.

2.3 Prétraitement

La notion de texture étant par définition relative à une impression d'ensemble, elle n'a de sens que sur une certaine étendue (on ne peut pas parler de texture au niveau du pixel). C'est pourquoi la classification doit se faire par l'intermédiaire d'une fenêtre d'observation, que l'on déplacera sur l'image. La fenêtre d'observation est une fenêtre carrée, dont la taille est égale à NxN pixels. Les paramètres sont calculés sur cette fenêtre et la classe trouvée est affectée au pixel central. Notons L(x,y) la luminance du pixel de coordonnées (x,y) dans la fenêtre. On définit ci-dessous les paramètres calculés.

Autocovariance :

$$C(\vec{\Delta}) = \frac{\sum_{y=y_a}^{y_b} \sum_{x=x_a}^{x_b} L^*(x,y)L^*(x+\delta_x, y+\delta_y)}{(y_b - y_a)(x_b - x_a)}$$

où

$$L^*(x,y) = L(x,y) - \eta$$

$$\eta = \frac{\sum_{y=0}^{N-1} \sum_{x=0}^{N-1} L(x,y)}{N^2} \quad (moyenne)$$

Les bornes x_a,x_b,y_a,y_b doivent être déterminées pour chaque déplacement $\vec{\Delta}(\delta_x, \delta_y)$ afin que celui-ci ne sorte pas de la fenêtre :

$$
\begin{aligned}
y_a &= sup\ (0, -\delta_y) \\
y_b &= inf\ (N-1, (N-1) - \delta_y) \\
x_a &= sup\ (0, -\delta_x) \\
x_b &= inf\ (N-1, (N-1) - \delta_x)
\end{aligned}
$$

Le calcul se fait jusqu'à une extension de 15 pixels dans les 2 dimensions. En exploitant la symétrie de l'autocovariance par rapport à l'origine, $C(\vec{\Delta})$ sera calculé pour tous les $\vec{\Delta}$ tels que:

$$
\begin{aligned}
-15 \leq \delta_y \leq 15 \qquad & 1 \leq \delta_x \leq 15 \\
1 \leq \delta_y \leq 15 \qquad & \delta_x = 0
\end{aligned}
$$

Une normalisation en amplitude est ensuite réalisée en divisant les $C(\vec{\Delta})$ par $C(\vec{0})$. On se ramène donc à des valeurs entre -1 et $+1$.

Une résolution égale au pixel pour les points éloignés de l'origine n'étant pas justifiée, on réalise un sous-échantillonnage comme indiqué sur la figure 2.

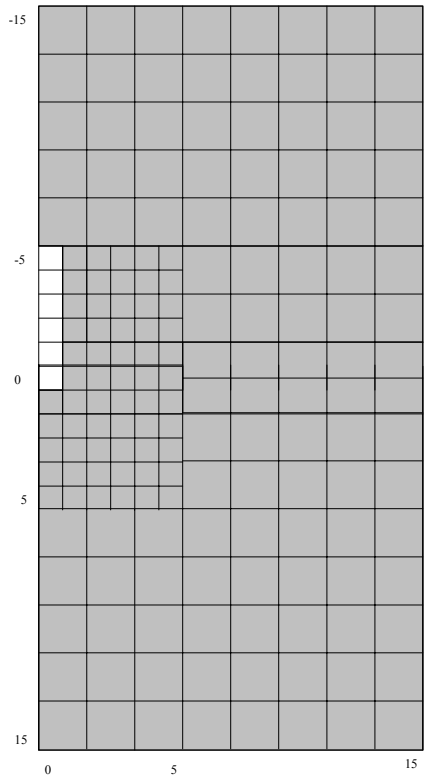


Figure 2: sous échantillonnage de l'autocovariance

Histogramme:

$$H(L) = \frac{nombre\ de\ pixels\ de\ luminance\ L}{N^2}$$

L'histogramme est ensuite centré sur sa valeur moyenne afin de ne pas être sensible à la luminance moyenne (par exemple, en Infra-Rouge, le ciel présente souvent un dégradé de luminance moyenne). Puis, il est sous-échantillonné avec un pas grandissant avec l'éloignement par rapport à la zone centrale (progression logarithmique), ce qui réduit l'influence du bruit, tout en conservant une bonne résolution dans la zone proche de la moyenne, qui est a priori la plus intéressante (fig 3).

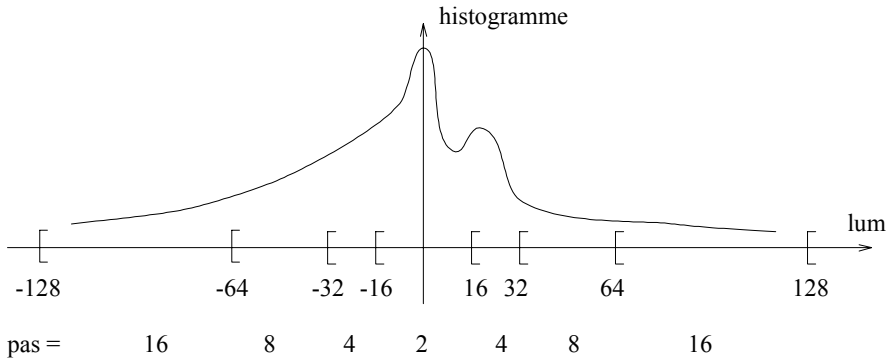


Figure 3: sous échantillonnage de l'histogramme

2.4 Le réseau de neurones

Nous utilisons un réseau de neurones multi-couches entrainé par l'algorithme de rétropropagation. Le système fonctionne en 2 étapes:

Première étape : La phase d'apprentissage

Durant cette phase, le réseau apprend à classifier correctement des exemples de chaque texture. L'algorithme d'apprentissage est l'algorithme de rétropropagation. Les exemples sont les paramètres (autocovariance + histogramme) calculés sur des fenêtres d'observation extraites des textures.

Seconde étape : La phase de reconnaissance

Le réseau peut maintenant classifier des images entières selon les textures qui y sont présentes. Ceci est réalisé par l'intermédiaire d'une fenêtre d'observation qui se déplace sur l'image. Pour chaque position, le réseau est alimenté par l'autocovariance et l'histogramme de la fenêtre. Les sorties du réseau sont calculées, et leur interprétation fournit une classe et une mesure de confiance. Lorsque toute l'image a été classifiée, un second passage permet de reclassifier les points dont la confiance est faible en effectuant un vote majoritaire pondéré dans leur voisinage.

Le réseau de neurones n'est pas directement alimenté par le contenu de la fenêtre d'observation car cela le forcerait à apprendre l'invariance en translation. Il est évident qu'une telle tâche est quasiment insurmontable car il serait nécessaire d'apprendre sur un très grand nombre d'exemples. L'histogramme et l'autocovariance présentent un double avantage: ils sont invariants en translation, et de plus ils caractérisent totalement la texture. Notons que l'autocovariance est par définition insensible à la valeur moyenne. De plus l'histogramme a également été normalisé par rapport à sa valeur moyenne. Ceci est souhaitable car la moyenne ne doit normalement pas être utilisée pour classifier la texture (si l'on change l'éclairage, la texture reste la même).

Le modèle du neurone est un sommateur pondéré suivi d'une non-linéarité. Le réseau utilisé est un perceptron multicouches (fig 4). Le réseau est entièrement connecté entre 2 couches successives, et est seuillé. La fonction d'activation du neurone est la tangente hyperbolique. Pour nos expérimentations, le nombre total d'entrées est 210 (40 pour l'histogramme et 170 pour l'autocovariance).

En sortie du réseau de neurones, on a un neurone par classe, et on utilise le codage suivant pour l'apprentissage:

classe 0 :	(1,-1,-1,...,-1,-1)
classe 1 :	(-1,1,-1,...,-1,-1)
...	...
classe n :	(-1,-1,-1,...,-1,1)

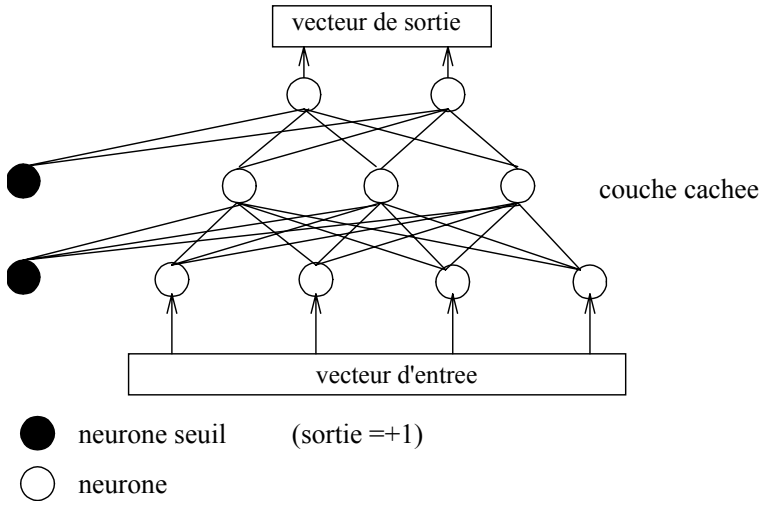


Figure 4: Perceptron multicouches

2.5 Calcul de la confiance et post-traitement

Pendant la phase de reconnaissance, le neurone dont la valeur de sortie est la plus forte détermine la classe. Notons j_1 ce neurone et O_{j_1} sa sortie. Une mesure de confiance est associée à la classification. La confiance est calculée comme:

$$CONFIANCE = min_{j \in sortie, j \neq j_1} \left(\frac{O_{j_1} - O_j}{2} \right)$$

Par exemple, pour quatre classes de textures, et les valeurs de sortie $(-1, +0.95, -0.35, -0.98)$, alors nous aurons:

$$\begin{aligned} CLASSE &= 1 \qquad (n^o \; du \; neurone \; qui \; a \; la \; plus \; forte \; sortie) \\ CONFIANCE &= \tfrac{1}{2} min\left((0.95 + 1.0), (0.95 + 0.35), (0.95 + 0.98)\right) \\ &= 0.65 \end{aligned}$$

La confiance est toujours une valeur entre 0 et 1, la dernière correspondant à une forte confiance. Le premier passage de la phase de reconnaissance produit une image de classification et une image de confiance. Puis un deuxième passage reclassifie les points dont la confiance est faible en effectuant un vote majoritaire pondéré par la distance dans un voisinage du point central. La nouvelle classe est obtenue en analysant les pixels voisins distants de d_{max} au plus (étendue de reclassification), afin de déterminer la classe la plus crédible.

Exemple de traitement pour 4 classes:

Soit un pixel(x,y) classifié 1 avec une confiance faible (inférieure à un seuil de 0.7 par exemple). On utilise un tableau d'accumulateurs C, la sommation étant réalisée su tous les pixels voisins à une distance d $\leq d_{max}$ du pixel(x,y)

$$\text{C(classe 0)}= \sum_{pixel \in classe0} k(d) confiance(pixel)$$

$$\text{C(classe 1)=confiance(x,y)+} \sum_{pixel \in classe1} k(d) confiance(pixel)$$

$$\text{C(classe 2)}= \sum_{pixel \in classe2} k(d) confiance(pixel)$$

$$\text{C(classe 3)}= \sum_{pixel \in classe3} k(d) confiance(pixel)$$

k(d) pondère l'influence de la confiance en fonction de la distance séparant le pixel à reclassifier et son voisin. Cette pondération suivra une loi gaussienne centrée ($\sigma = 7.5$ pour nos expérimentations):

$$k(d) = e^{-(d^2/2\sigma^2)}$$

A l'issue, de ce traitement on recherche le maximum du tableau C(classe). Soit C_{max}(classe i) ce maximum. On a alors:

$$\text{nouvelle classe} = \text{classe i}$$

3 Résultats statistiques

3.1 Conditions expérimentales

Les résultats statistiques sont fournis pour des images Infra-Rouge [*Jacquet*91] provenant des sites N (site du Nord de la France) et S (site du Sud de la France).

Sur le site N, 3 classes ont été définies: ciel (0), terre (1), et habitations (2). Sur le site S, 2 classes ont été définies: ciel (0), et terre (1). Pour chaque site et chaque classe, une image 512x512 contenant 64 imagettes 64x64 a été crée (fig 6 à 8). Ces imagettes représentent les textures par différentes conditions météorologiques, différentes heures du jour et de la nuit, et pour différentes zones du panorama.

La discrimination entre les différentes textures n'est pas évidente, même pour l'observateur humain. Les textures "ciel" et "terre" sont particulièrement difficile à différencier.

Ceci permet une évaluation statistique de la méthode. La fenêtre d'observation utilisée a une taille de 31x31 pixels et l'autocovariance dans cette fenêtre est calculée avec une extension de 15 pixels. L'autocovariance est sous-échantillonnée d'un facteur linéaire 2 pour les zones éloignées du centre. L'histogramme est également sous-échantillonné. Les bases d'apprentissage et d'évaluation ont été créés de la façon suivante:

Base d'apprentissage : Les 32 blocs d'ordre pair de chaque image ont été utilisés. Pour chaque bloc, 10 fenêtres d'observation de 31x31 pixels ont été choisies aléatoirement. On a donc 320 exemples par classe, soit un total de 960 exemples pour le site N et 640 exemples pour le site S.

Base d'évaluation : Les 32 blocs d'ordre impair de chaque image ont été utilisés. Pour chaque bloc, 5 fenêtres d'observation de 31x31 pixels ont été choisies aléatoirement. On a donc 160 exemples par classe, soit un total de 480 exemples pour le site N et 320 exemples pour le site S.

3.2 Résultats statistiques et comparaisons

Nous donnons ci-dessous les taux de reconnaissance obtenus (sans post-traitement). A titre de comparaison, les résultats obtenus avec d'autres classifieurs (plus-proches-voisins, pseudo-inverse, méthode de Hebb) alimentés par les mêmes paramètres sont également fournis. Les méthodes sont classées par ordre de taux de généralisation décroissant. Les matrices de confusion correspondant au meilleur classifieur sont également fournies. Nous donnerons ensuite les résultats du traitement d'images complètes.

Site N

Méthode	Apprentissage	Généralisation	Nb de multiplications
Réseau 3 couches	96.6%	86.5%	1278
Réseau 2 couches	92.0%	84.0%	630
Plus_Proche_Voisin	100.0%	72.9%	201600
Pseudo_Inverse	100.0%	72.7%	630
5-Plus_Proches_Voisins	97.0%	68.5%	201600
3-Plus_Proches_Voisins	97.5%	68.3%	201600
Hebb	45.7%	41.9%	630

Le réseau à 2 couches contient 210+3 neurones, et le réseau à 3 couches 210+6+3 neurones (neurones seuils non compris).

Matrice de confusion pour le réseau à 3 couches (210+6+3 neurones)

Classe	Effectif	Matrice de confusion		
0	320 (33.3%)	86.9	9.4	3.7
1	320 (33.3%)	13.8	82.5	3.7
2	320 (33.3%)	0.6	9.4	90.0
Taux moyen de généralisation : **86.5%**				

Site S

Méthode	Apprentissage	Généralisation	Nb de multiplications
Réseau à 3 couches	97.5%	93.7%	848
Réseau à 2 couches	97.5%	93.4%	420
5-Plus-Proches-Voisins	94.7%	88.1%	134400
3-Plus-Proches-Voisins	97.3%	87.5%	134400
Plus-Proche-Voisin	100.0%	85.6%	134400
Hebb	79.7%	78.4%	420
Pseudo-Inverse	98.9%	73.8%	420

Le réseau à 2 couches contient 210+2 neurones, et le réseau à 3 couches 210+4+2 neurones (neurones seuils non compris).

Matrice de confusion pour le réseau à 3 couches (210+4+2)

Classe	Effectif	Matrice de confusion
0	160 (50.0%)	90.6 9.4
1	160 (50.0%)	3.1 96.9
Taux moyen de généralisation : **93.8%**		

4 Traitement d'images complètes

Afin d'illustrer les résultats obtenus sur le traitement d'images complètes nous présentons la classification d'images Infra-Rouge des sites N et S. Le réseau utilisé est un réseau à 3 couches (210+6+3 neurones). Notre méthode est également valide dans le domaine "visible", comme le prouvent les résultats obtenus sur les textures de Brodatz [*Brodatz*66] et l'image aérienne que nous présentons ci-dessous.

- **Textures de Brodatz:**

  La figure 10 illustre le résultat obtenu pour le traitement d'images de Brodatz, qui servent souvent de référence dans le domaine de la discrimination de textures. La fenêtre d'observation pour l'analyse est de 32x32 pixels. L'apprentissage a été réalisé sur les textures de synthèse (dont on rappelle qu'elles ont été créées à partir des paramètres extraits sur des fenêtres 64x64 de chaque texture). On a pris 100 exemples par texture.

  Sur l'image on a représenté la source (en haut à gauche), la classification intermédiaire (en haut à droite), la confiance (en bas à gauche), et la classification définitive (en bas à droite).

- **Image aérienne:**

  La figure 11 illustre le résultat obtenu pour le traitement d'une image aérienne. Les fenêtres incrustées dans l'image source indiquent les exemples d'apprentissage. La taille de ces fenêtres est de 25x25 pixels pour obtenir une meilleure précision sur les frontières. Des exemples supplémentaires ont été obtenus en déplaçant ces fenêtres de 1 à 4 pixels dans leur voisinage. Le

nombre total d'exemples par texture est donc de 81. L'autocorrélation n'a ici été calculée que sur une extension de 7 pixels.

- **Images Infra-Rouge:**

  Les figures 12 et 13 illustrent la classification d'images Infra-Rouge. L'apprentissage n'a pas été réalisé sur ces images, mais sur d'autres zones du panorama pour diverses conditions météorologiques. Sur la figure 12, on distingue en haut à gauche l'image source (détramée pour éviter le flou dû au mouvement du capteur), et en bas à droite la classification définitive. Sur la figure 13, on distingue en haut à gauche une trame de l'image source, et en bas à droite la classification correspondante.

  Cette classification est globalement satisfaisante, hormis sur les frontières entre zones (une approche multi-résolution avec plusieurs tailles de fenêtres permettrait d'améliorer la précision aux frontières). Il y a peu d'erreurs de classification sur le ciel malgré le fort dégradé de luminance moyenne (dû à la variation de la température avec l'altitude) et la présence de nuages sur l'horizon de la figure 12. Certaines erreurs de classification sur les bords des images sont dûes à des problèmes de digitalisation des images, et non pas au système de classification.

5 Conclusion

Nous avons montré la faisabilité d'une approche basée sur la conjecture de Gagalowicz pour la classification de fonds. L'étude comparative avec d'autres classifieurs montre un net avantage pour le classifieur connexionniste (86.5% contre 72.9% avec le plus proche voisin sur le site N).

Il resterait à comparer le prétraitement que nous proposons avec d'autres prétraitements. Il est possible que sur une base de données réduite d'autres attributs soient aussi performants que ceux que nous utilisons (autocovariance + histogramme). Mais notre prétraitement a l'avantage de conserver toute l'information pertinente, ainsi que nous l'avons montré par synthèse. Il n'y a donc pas de risques de recouvrement de classes dans l'espace des paramètres lorsque la base de données devient importante . Cet élément ne doit pas être négligé car un système opérationnel devra travailler sur un volume important de données, et être insensible aux conditions météorologiques, à l'éclairage, etc.

La mise en œuvre du système est très simple car il fonctionne par apprentissage. De plus, les opérations réalisées, tant au niveau du prétraitement que du réseau de neurones sont simples, ce qui conduit à penser qu'une réalisation matérielle ne devrait pas poser de problème majeur. Il serait toutefois souhaitable d'étudier la possibilité de réduire la charge de calcul en exploitant diverses redondances, notamment au niveau du calcul de l'autocovariance.

De nombreuses amélioration restent possibles. Par exemple, l'utilisation de poids partagés et de connexions locales dans le réseau devrait améliorer les performances en généralisation. En effet, la structure bi-dimensionnelle de l'autocovariance n'est pas du tout traduite dans les connexions du réseau utilisé actuellement. D'autre part, une approche multi-résolution, avec plusieurs tailles de fenêtre permettrait d'améliorer la précision sur les frontières.

Références

[*Ahuja*81] N. AHUJA, A. ROSENFELD,
"Mosaic models for textures",
IEEE PAMI, vol 3, no 1, January 1981

[*Bovick*90] A.C. BOVIK, M. CLARK, W.S. GEISLER,
"Multichanel texture analysis using localized spatial filters",
IEEE PAMI, vol 12, no 1, January 1990

[*Brodatz*66] P. BRODATZ
"Textures - A Photographic Album for Artists and Designers"
Dover Publications Inc. , New York, 1966

[*Chen*83] P.C. CHEN, T. PAVLIDIS
"Segmentation by texture using correlation"
IEEE PAMI, vol 5, no 1, January 1983

[*Clark*87] M. CLARK, A.C. BOVIK, W.S. GEISLER
"Texture segmentation using Gabor modulation/demodulation"
Pattern recognition Letters 6 (1987) 261-267, September 1987

[*Coggins*85] J.M. COGGINS, A.K. JAIN
"A spatial filtering approach to texture analysis"
Pattern Recognition Letters 3 (1985) 195-203, May 1985

[*Conners*80] R.W. CONNERS, C.A. HARLOW
"A theoretical comparison of texture algorithms"
IEEE PAMI, vol. PAMI-2, no. 3, May 1980

[*Cross*83] G.R. CROSS, A.K. JAIN
"Markov Random Field Texture Models"
IEEE PAMI , vol 5, no 1, January 1983

[*Davis*79] L.S. DAVIS, S.A. JOHNS, J.K. ARGAWAL
"Texture analysis using generalized co-occurence matrices"
IEEE PAMI, vol 1, no 3, July 1979

[*Derin*86] H. DERIN, W.S. COLE
"Segmentation of textured images using Gibbs Random Fields"
Computer Vision, Graphics, and Image Processing 35, 72-98 (1986)

[*Derin*87] H. DERIN, H. ELLIOTT,
"Modeling and segmentation of noisy and
textured images using Gibbs Random Fields"
IEEE PAMI, vol 9, no 1, January 1987

[*Eichmann*88] G. EICHMANN, T. KASPARIS,
"Topologically invariant texture descriptors",
Computer Vision, Graphics, and Image Processing 41, 267-281 (1988)

[*Faugeras*80] O.D. FAUGERAS, W.K.PRATT,
"Decorrelation methods of texture feature extraction"
IEEE PAMI, vol 2, no 4, July 1980

[*Gagalowicz*80] A. GAGALOWICZ,
"Visual discrimination of stochastic texture fields
based upon their second order statistics"
ICPR - Miami - Dec 1980

[*Gagalowicz*83] A. GAGALOWICZ
"Vers un modèle de textures"
Thèse de doctorat d'état ès sciences mathématiques
Université de PARIS VI . 1983

[*Haralick*73] R.M. HARALICK, K. SHANMUGAM, I. DINSTEIN,
"Textural features for image classification",
IEEE Trans. Syst.,Man,Cybern., vol SMC-3, pp.610-621, Nov. 1973

[*Haralick*79] R.M. HARALICK
"Statistical and structural approaches to texture"
Proc. IEEE, vol 67, no 5, May 1979

[*Henocq*90] Hughes HENOCQ
"Synthèse de textures et classification par réseaux de neurones"
Rapport de stage DEA d'Electronique - Brest - Juillet 1990
Thomson CSF/LER - Responsable de stage G. BUREL

[*Holmes*84] Q.A. HOLMES, D.R. NÜESCH, R.A. SHUCHMAN
"Textural analysis and real-time classification
of sea-ice types using SAR data"
IEEE Trans. on Geoscience and Remote Sensing, vol GE-22, no 2, March 1984

[*Hsiao*89] J.Y. HSIAO, A.A. SAWCHUK,
"Supervised textured image segmentation using feature smoothing
and probabilistic relaxation techniques"
IEEE PAMI, vol 11 , no 12, December 1989

[*Jacquet*91] Florence JACQUET, Gilles BUREL, et al.
"Application des réseaux de neurones à la Veille Panoramique Infra-Rouge"
Revue Technique THOMSON CSF, vol 23, n^o1, mars 1991

[*Jernigan*84] M.E. JERNIGAN, F. D'ASTOUS,
"Entropy-based texture analysis in the spatial frequency domain",
IEEE PAMI, vol 6, no 2, March 1984

[*Kashyap*86] R.L. KASHYAP, A. KHOTANZAD,
"A model-based method for rotation invariant texture classification",
IEEE PAMI, vol 8, no 4, July 1986

[*Leu*85] J.G. LEU, W.G. WEE
"Detecting the spatial structure of natural textures based on shape analysis"
Computer Vision, Graphics, and Image Processing 31 , 67-88 (1985)

[*Lumia*83] R. LUMIA, R.M. HARALICK, et al.
"Texture analysis of aerial photographs"
Pattern Recognition, vol 16, no 1, pp 39-46, 1983

[*Reed*90] T.R. REED, H. WECHSLER,
"Segmentation of textured images and gestalt organization
using spatial/spatial-frequency representations"
IEEE PAMI, vol 12, no 1, January 1990

[*Ronsin*85] J.RONSIN, D. BARBA, S. RABOISSON,
"Comparison between co-occurence matrices, local histograms,
and curvilinear integration for texture characterization"
SPIE Symposium vol 596, Cannes 1985

[*Sun*83] C. SUN, W.G. WEE,
"Neighboring gray level dependence matrix for texture classification",
Computer Vision, Graphics, and Image Processing 23, 341-352 (1983)

[*Tomita*82] F. TOMITA, Y. SHIRAI, S. TSUJI,
"Description of textures by structural analysis",
IEEE PAMI, vol 4, no 2, March 1982

[*Tüceryan*90] M. TÜCERYAN, A.K. JAIN,
"Texture segmentation using Voronoi polygons",
IEEE PAMI, vol 12, no 2, February 1990

[*Vickers*82] A.L. VICKERS, J.W. MODESTINO
"A maximum likehood approach to texture classification"
IEEE PAMI, vol 4, no 1, January 1982

[*Vilnrotter*86] F.M. VILNROTTER, R. NEVATIA, K.E. PRICE,
"Structural analysis of natural textures",
IEEE PAMI, vol 8, no 1, January 1986

[*Wechsler*79] H. WECHSLER, M. KIDODE,
"A random walk procedure for texture discrimination",
IEEE PAMI, vol 1, no 3, July 1979

[*Wechsler*80] H. WECHSLER,
"Texture Analysis - A survey",
Signal Processing 2 (1980) 271-282

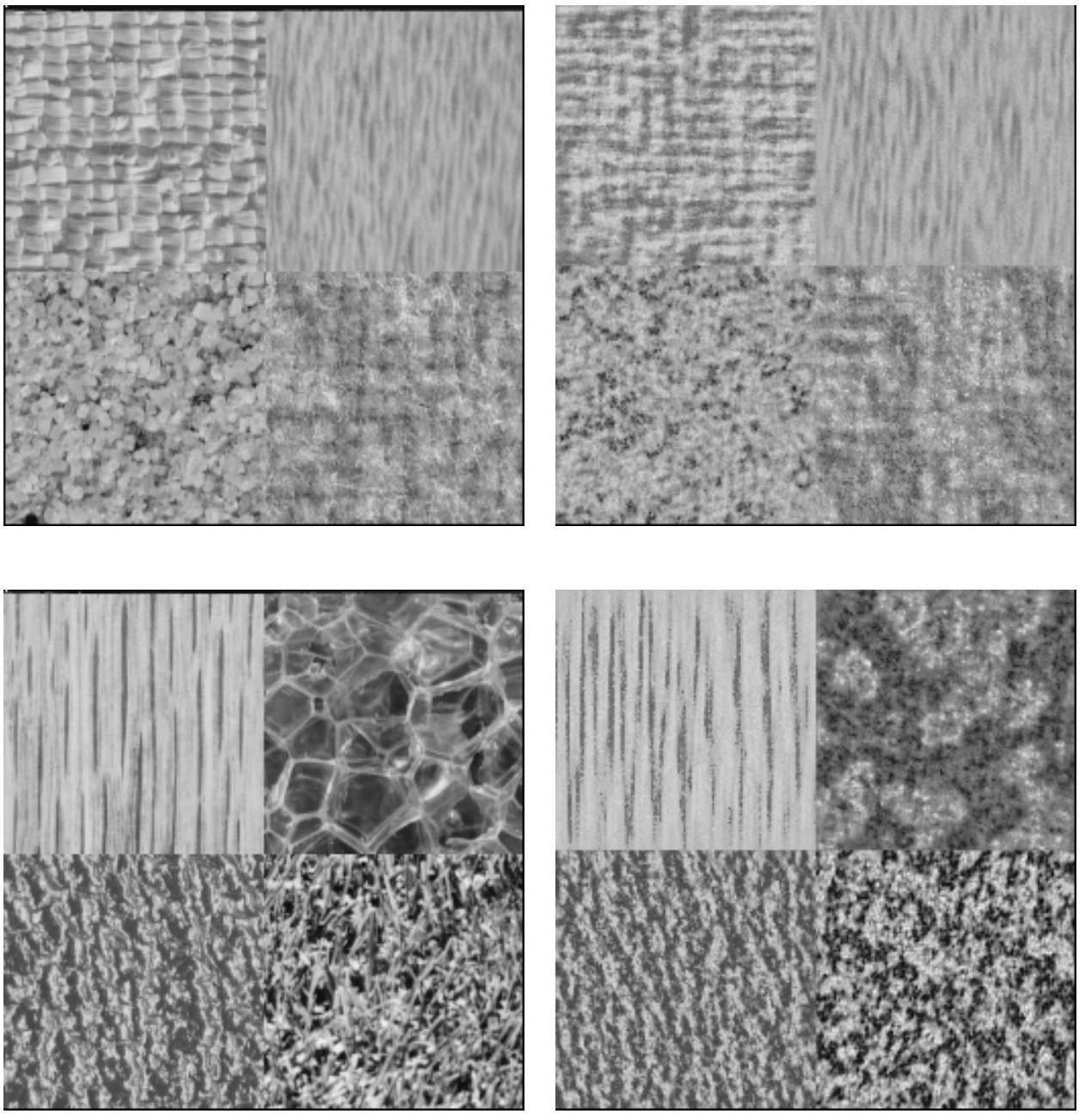


Figure 5: Textures de Brodatz (à gauche) et leur synthèse (à droite)

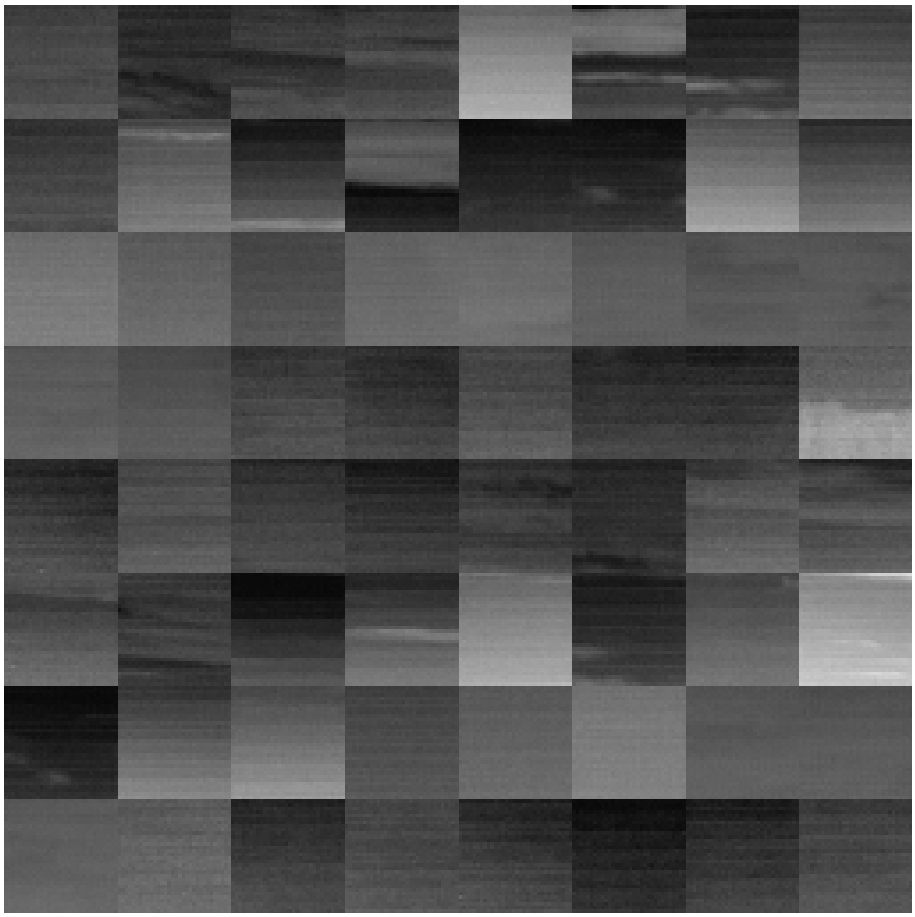


Figure 6: Imagettes de la texture “ciel”

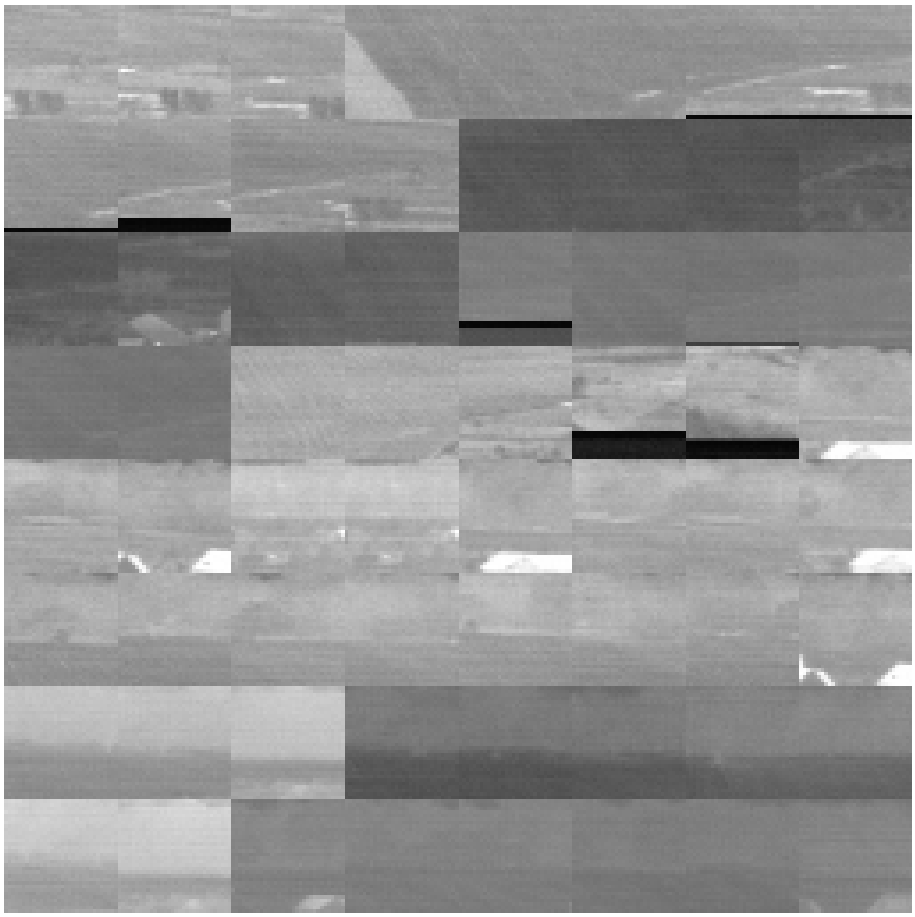


Figure 7: Imagettes de la texture “terre”

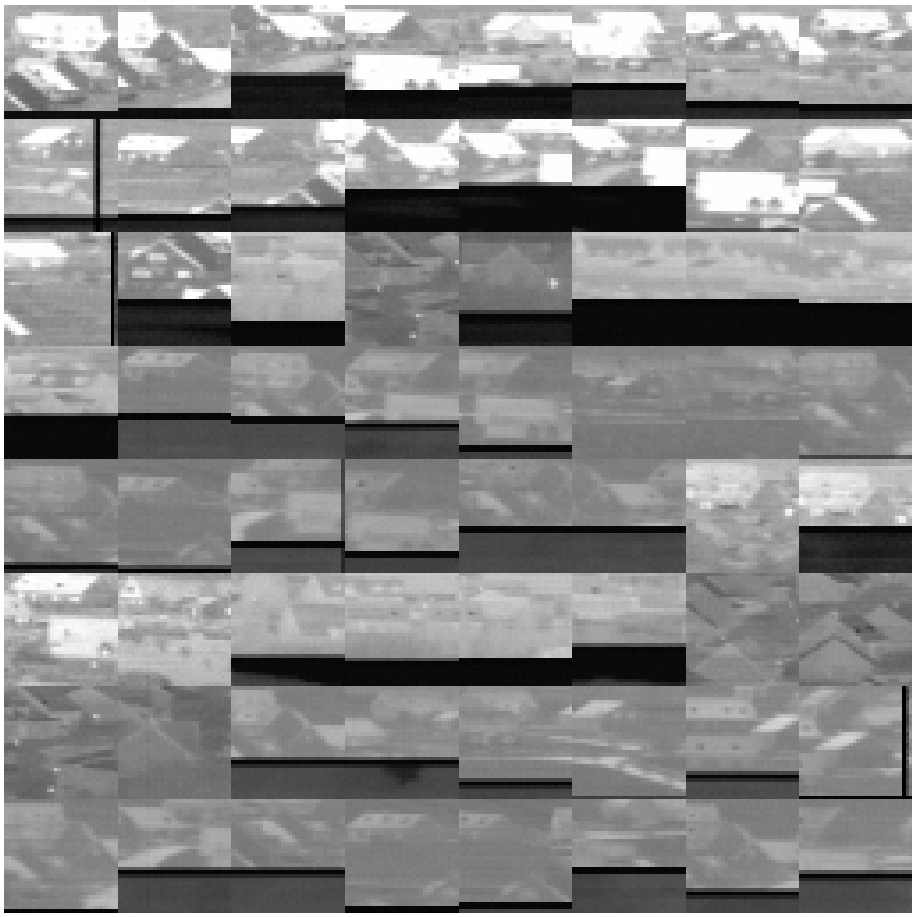


Figure 8: Imagettes de la texture “habitations”

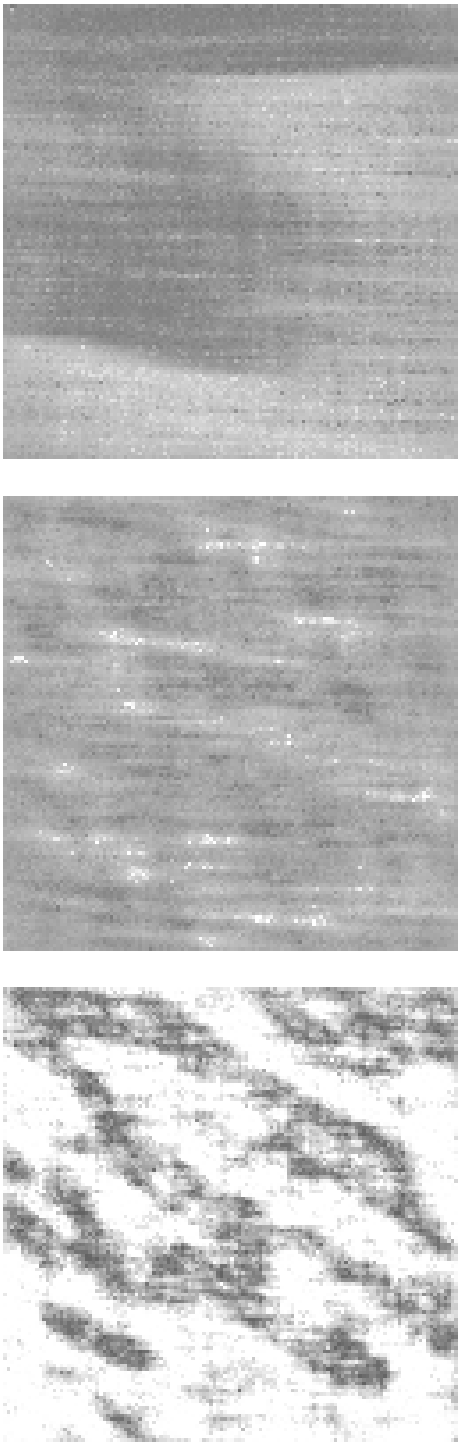


Figure 9: Synthèse de textures Infra-Rouge (ciel, terre, et habitations)

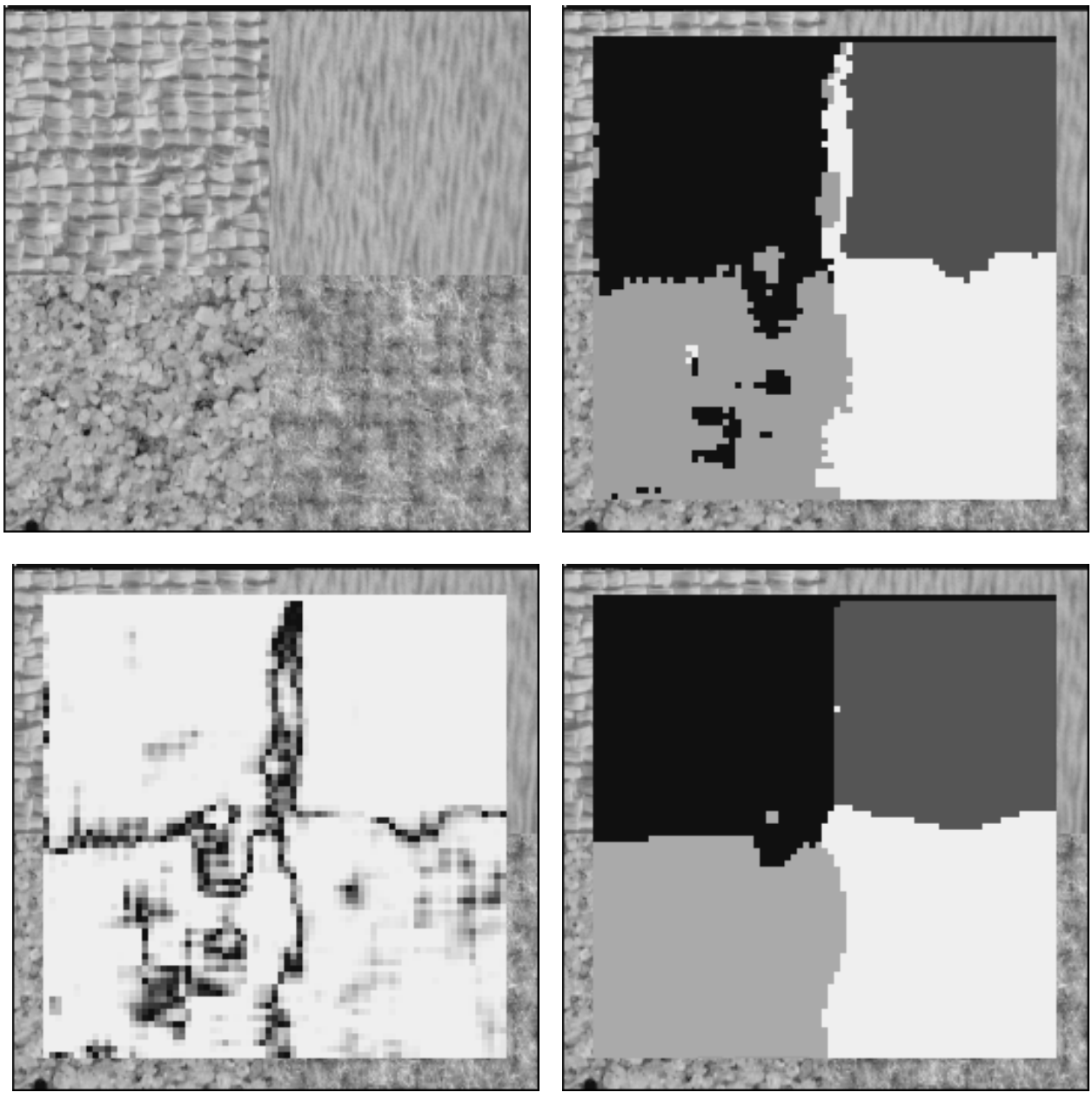


Figure 10: Classification de textures de Brodatz

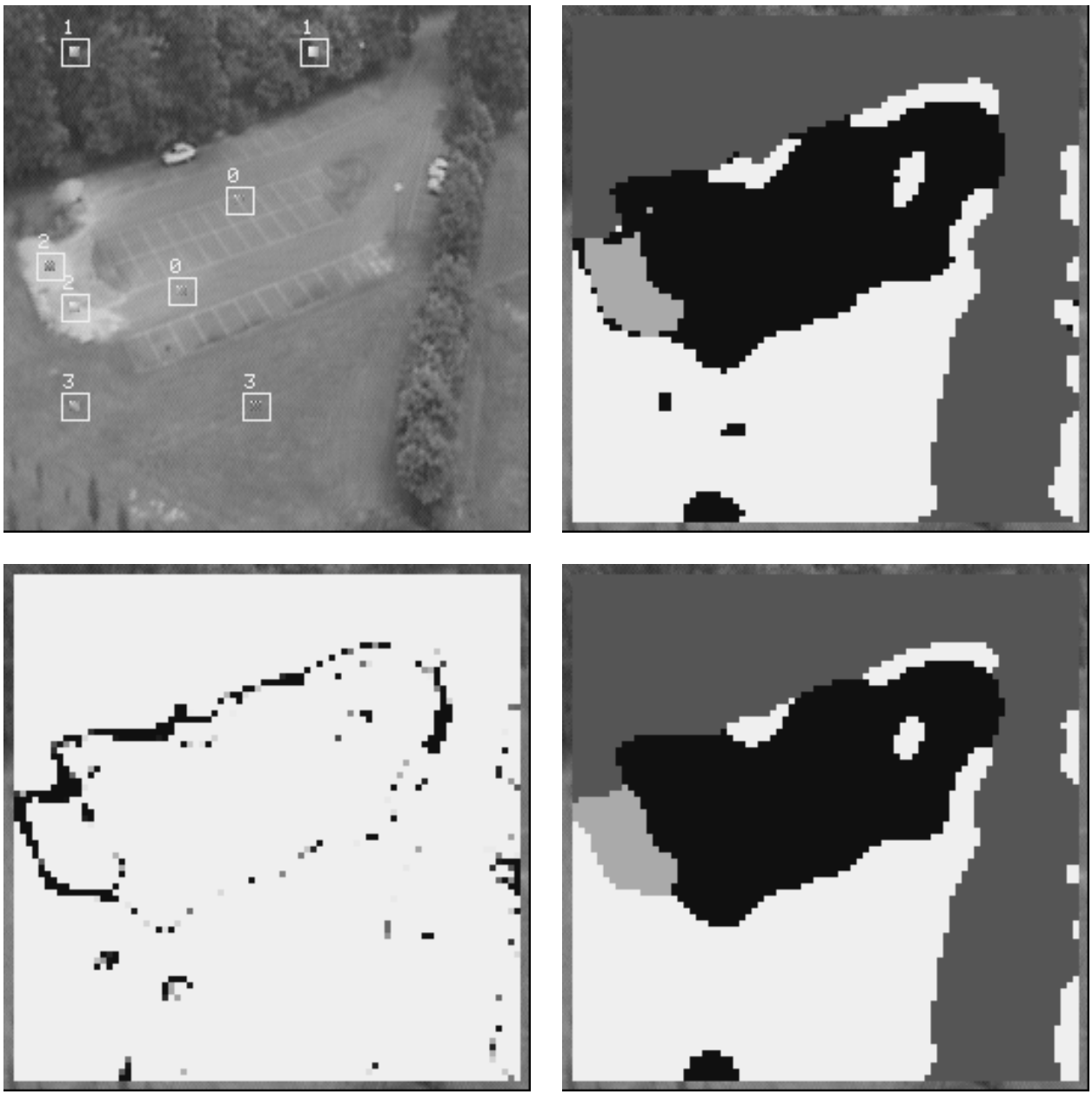


Figure 11: Classification sur image aérienne

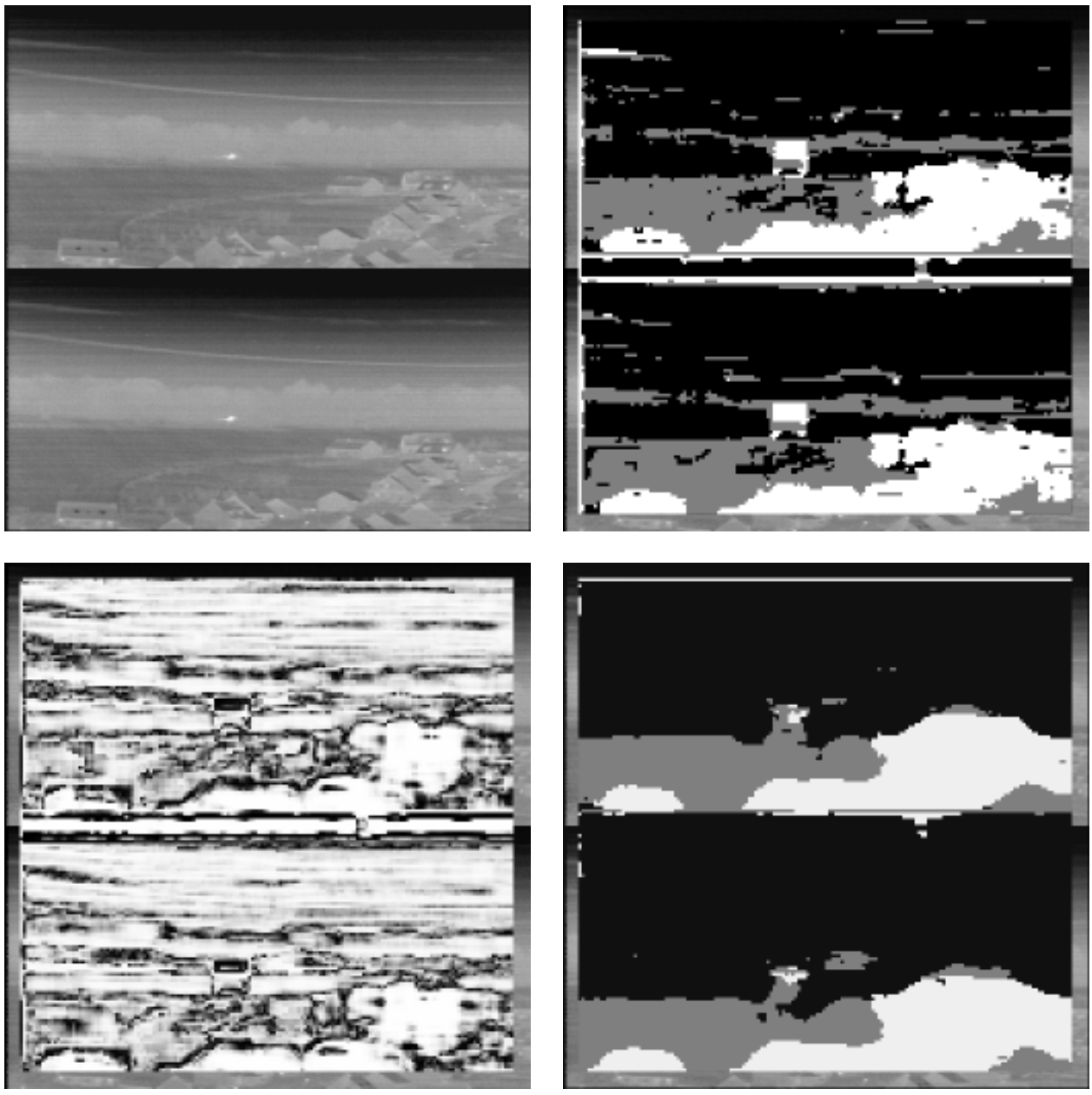


Figure 12: Classification sur le site N

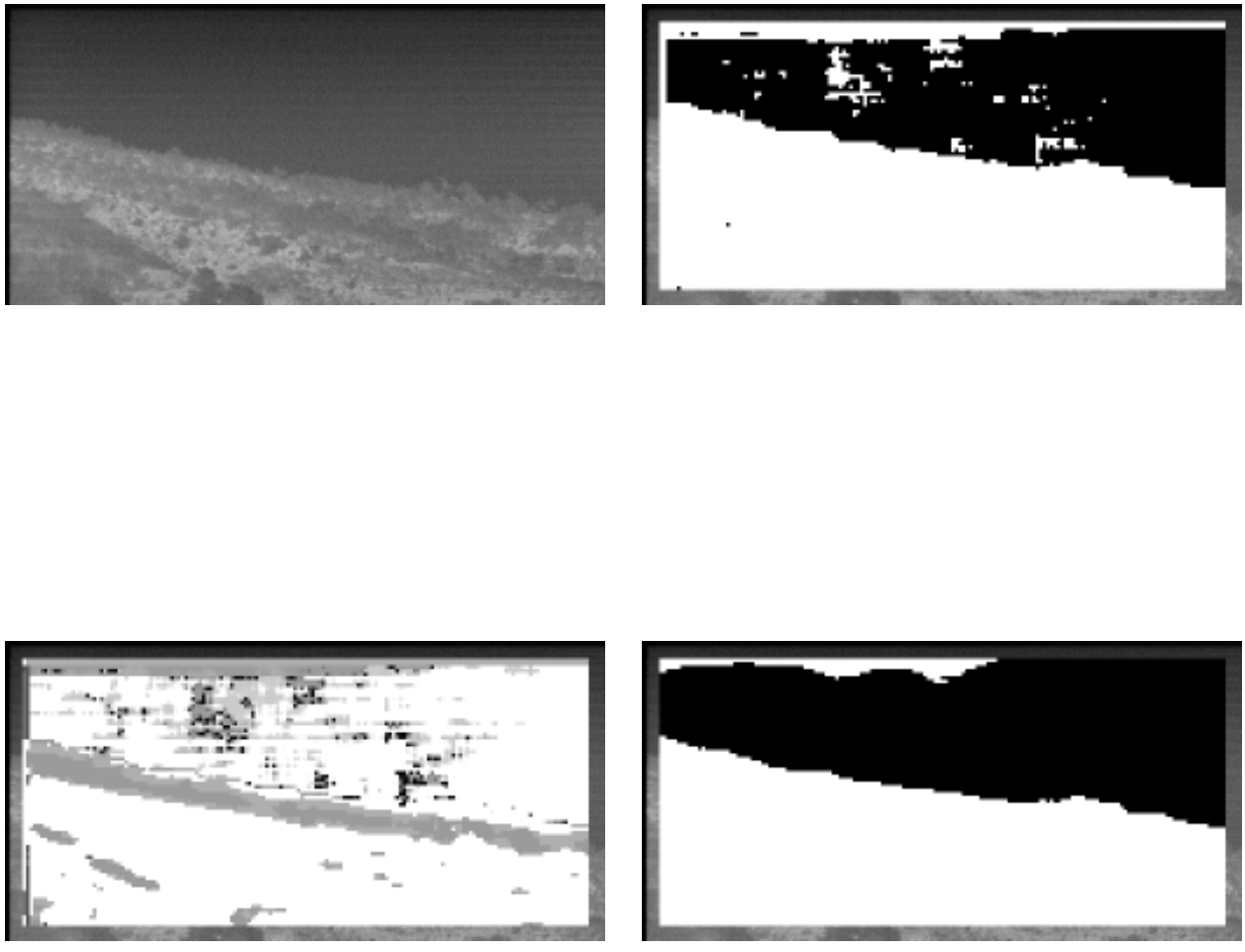


Figure 13: Classification sur le site S

Chapitre 6:

RECONNAISSANCE D'OBJETS 2D

Nous proposons un système connexionniste pour la reconnaissance d'objets partiellement cachés (pièces industrielles 2D). La stratégie choisie est la suivante: à partir d'un contour de l'objet à reconnaître, une approximation polygonale est réalisée. Puis, des fonctions de reconnaissance sont calculées sur le résultat de l'approximation polygonale (typiquement, une fonction de reconnaissance est une fonction logique qui est vraie lorsque des configurations angulaires spécifiques sont trouvées sur le polygone). Les valeurs de ces fonctions alimentent un réseau de neurones multi-couches entraîné par l'algorithme de rétropropagation (utilisant les améliorations décrites au chapitre 3).

Pendant la phase d'apprentissage, on entraîne le réseau de neurones à reconnaître des objets partiellement cachés (les configurations d'occlusion sont choisies aléatoirement). Puis, le réseau est testé sur des images qui n'ont pas servi à l'apprentissage. Des expérimentations conduites sur une centaine d'images, avec environ 5 objets (souvent partiellement cachés) par image ont montré un taux de reconnaissance de 96%. Nous proposons également une comparaison avec un système d'apprentissage symbolique afin de mettre en valeur les avantages et les inconvénients de chaque méthode.

Les travaux antérieurs sont résumés dans le paragraphe 1. Notre méthode est détaillée dans le paragraphe 2. Les paragraphes 3 et 4 décrivent respectivement un système symbolique et un classifieur Bayésien que nous avons utilisés pour remplacer le réseau de neurones, à des fins de comparaison. Les résultats expérimentaux sont fournis dans le paragraphe 5.

1 Travaux antérieurs

1.1 Généralités

Un pré-requis important pour le développement de l'automatisation industrielle est la faculté pour un robot de reconnaître et de localiser des pièces industrielles par le biais de la vision. Ainsi que l'ont noté Bolles[*Bolles*82] et Ayache[*Ayache*86], la plus grande partie des systèmes industriels de vision, tels que le "VS-100 Vision System" de Machine Intelligence Corporation (1981), est limitée à la reconnaissance et la localisation d'objets isolés contrastant avec le fond. Pourtant, il existe nombre de tâches importantes pour lesquelles il est impossible, ou trop coûteux, de disposer les objets de telle sorte qu'ils soient isolés.

Récemment, des méthodes pour localiser et reconnaître des objets qui sont seulement partiellement visibles ont été proposées. Ces méthodes, qui sont encore surtout du domaine du laboratoire, sont généralement basées sur la détection de caractéristiques locales (coins, trous, ...), suivie par un système de génération/vérification d'hypothèses. Ce système cherche à mettre en correspondance les caractéristiques détectées sur l'image avec celles des modèles. Certaines de ces méthodes sont performantes, mais elles sont généralement coûteuses en puissance de calcul et en temps CPU, car les hypothèses (correspondances) générées sont rarement correctes dès les premières itérations, et de nombreux retours en arrière sont donc nécessaires. De plus, l'intervention d'un opérateur est souvent requise pour sélectionner les caractéristiques intéressantes et pour construire la base de modèles.

Nous ne passerons pas en revue ces diverses méthodes, qui sont finalement assez voisines dans leur principe, et diffèrent surtout par les caractéristiques locales choisies (par exemple Bolles [*Bolles*82] utilise les coins et les trous, alors que Koch [*Koch*87] utilise des couples de segments), et la façon dont les contraintes, issues de la rigidité des objets, sont mises à profit pour éviter l'explosion combinatoire.

Nous décrivons brièvement la méthode "HYPER" (HYpotheses Predicted and Evaluated Recursively) proposée par Ayache et Faugeras [*Ayache*86], d'une part parce qu'elle est bien représentative des méthodes à base d'hypothèses [*Bolles*82][*Koch*87][*Simon*88][*Grimson*87], et d'autre part parce qu'elle semble être l'une des plus efficaces et des mieux justifiées. Nous dirons également un mot des méthodes par transformée de Hough [*Turney*84][*Rives*85], bien que ce type de méthode semble être plus marginal.

1.2 La méthode "HYPER"

Description de la scène et des modèles

L'objectif de la méthode est de reconnaître et localiser des objets 2D avec possibilité de recouvrements et de mauvaises conditions d'éclairage. On suppose approximativement connu le facteur d'échelle. La scène est décrite sous forme d'un ensemble de segments (obtenus après extraction de contour et approximation polygonale). Les modèles sont également décrits sous cette forme (typiquement 50 à 100 segments par modèle). Les dix segments les plus longs d'un modèle seront nommés "segments privilégiés".

Génération d'hypothèses

Une hypothèse est la mise en correspondance d'un segment privilégié avec un segment de la scène. On a typiquement de l'ordre de quelques centaines d'hypothèses possibles par modèle. Afin d'éviter d'examiner des hypothèses peu vraisemblables, on impose les contraintes suivantes. Notons M_0 le segment du modèle et S_0 le segment de la scène auquel on l'associe. Une hypothèse doit être telle que la longueur de S_0 ne diffère pas de plus de 30% de celle de M_0. De plus, l'angle que fait M_0 avec son prédécesseur (sur le polygone) ne doit pas différer de plus de 30^o de l'angle que fait S_0 avec son prédécesseur.

On a typiquement de l'ordre de quelques centaines d'hypothèses possibles par modèle, que l'on classe par ordre de compatibilité décroissante au niveau de l'angle avec le prédécesseur. Ceci va permettre d'évaluer en priorité les hypothèses les plus vraisemblables suivant ce critère.

Evaluation d'hypothèse

L'hypothèse (M_0, S_0) définit un transformation initiale que l'on peut représenter par un vecteur $T = [k.cos\theta, k.sin\theta, tx, ty]$, où θ désigne la rotation entre l'objet et le modèle, k le facteur d'échelle, tx et ty la translation. A partir d'une hypothèse (M_0, S_0), on va chercher à mettre en correspondance de nouveaux segments $M_1, M_2, M_3, ...$ du modèle avec des segments de la scène. On traite en priorité les segments proches de M_0 (fig 1).

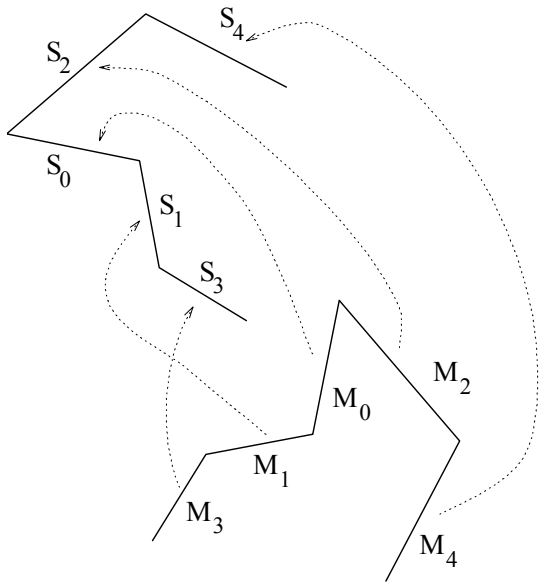


Figure 1: Ordre d'examen des segments dans la méthode "HYPER"

Soit un segment M_i du modèle, et notons $M_i^* = T(M_i)$ son image dans la scène. Pour tout segment S_j de la scène, on calcule une distance d_{ij} entre M_i^* et S_j (somme pondérée de l'écart en longueur, de l'écart angulaire, et de la différence de position du point central). M_i sera mis en correspondance avec le segment S_j qui produit le d_{ij} minimal. Ceci sous réserve que les écarts en angle, longueur et position ne dépassent pas certains seuils, auquel cas M_i est mis en correspondance avec NIL (ensemble vide).

A chaque nouvelle mise en correspondance, on met à jour la transformation T en minimisant $R = \sum_i l_i \|m_i^* - s_i\|^2$ où m_i^* et s_i sont les points centraux de $T(M_i)$ et de S_i, et l_i est la longueur de M_i (on privilégie les longs segments). L'indice i prend pour valeurs les indices des segments déjà mis en correspondance. Cette minimisation est réalisée récursivement en utilisant un filtre de Kalman.

On met également à jour, à chaque nouvelle mise en correspondance, une mesure de qualité Q qui est le rapport entre la somme des longueurs des segments identifiés et la somme des longueurs de tous les segments du modèle. Lorsque l'on a examiné tous les segments du modèle, on obtient une transformation T_N et une mesure de qualité Q_N (N étant le nombre de segments du modèle).

Fin du processus

Le processus s'arrête lorsque le nombre d'hypothèses examinées est assez large (de l'ordre de quelques dizaines, sachant que l'on examine les hypothèses dans l'ordre dans lequel elles ont été rangées), ou lorsqu'une très bonne mesure de qualité est atteinte. Ensuite la meilleure hypothèse (celle qui a produit le Q_N maximum) est réexaminée en initialisant T par la transformation T_N et en refaisant tourner le filtre de Kalman. Ceci permet d'obtenir une meilleure estimation de la transformation (car on part d'une meilleure estimation initiale), et éventuellement de mettre en correspondance de nouveaux segments. La qualité est remise à jour et l'hypothèse est finalement conservée si la qualité est supérieure à un seuil (typiquement 0.25). On recommence ensuite l'ensemble du processus avec les autres modèles.

1.3 Méthodes par transformée de Hough

Turney [*Turney*84] a proposé une méthode à base de "template matching" (mise en correspondance de structures élémentaires) et de transformée de Hough. L'objectif est toujours de reconnaître et de localiser des objets 2D avec possibilité de recouvrements et de mauvaises conditions d'éclairage. On suppose parfaitement connu le facteur d'échelle.

Choix des structures élémentaires

On recherche sur le contour du modèle de l'objet des portions de contour qui sont bien discriminantes (fig 2). Une mesure d'efficacité (e) a été définie pour guider ce choix. Puis on représente ces portions de contour en $\theta - s$, c.a.d. angle de la tangente au contour en fonction de l'abscisse curviligne. A chaque portion du contour est également associé un vecteur $\vec{w}$ qui part du milieu de la portion de contour et qui pointe vers le centre de gravité de l'objet.

Mise en correspondance

On extrait les contours de la scène, et l'on représente les contours connectés en $\theta - s$. Puis on effectue un balayage avec la structure élémentaire recherchée (fig 3). Pour

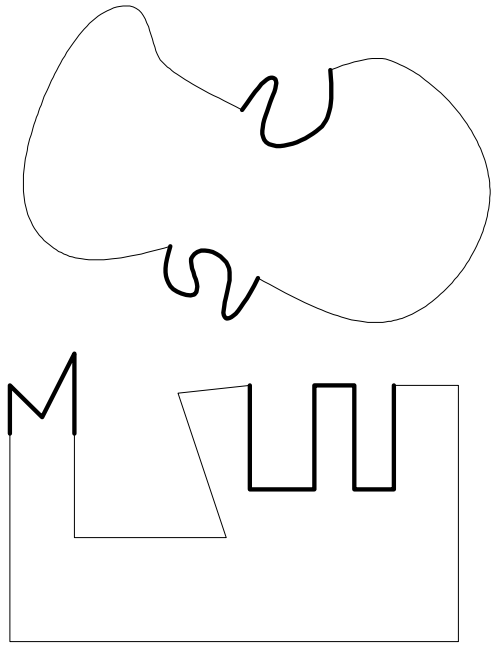


Figure 2: Exemples de structures élémentaires (méthode de Turney)

chaque valeur de s, une mesure d'erreur est calculée:

$$\gamma(s) = \int_t [\theta_{template}(t) - \theta_{contour}(t+s) + \theta_0]^2 \, dt$$

on détermine la meilleure valeur $\gamma^*(s)$ en faisant varier θ_0, et on définit un coefficient de qualité de la correspondance:

$$c(s) = \frac{1}{1+\gamma^*(s)}$$

Transformée de Hough

Pour chaque valeur de s, on vote pour le point G vers lequel pointe le vecteur $\vec{w}$ avec un poids $c(s).e$ (où e est l'efficacité de la structure élémentaire). L'angle θ_0^* qui a produit $\gamma^*(s)$ est ajouté à une liste d'angles liés à ce point. Si à la fin du processus, un point G a recueilli assez de votes, on moyenne les angles qui lui ont été associés pour trouver l'orientation de l'objet.

D'après [*Turney*84], la durée de l'ensemble du processus pour une scène de difficulté moyenne est de 20mn CPU par objet sur un VAX11/780 (le programme ayant été écrit en C).

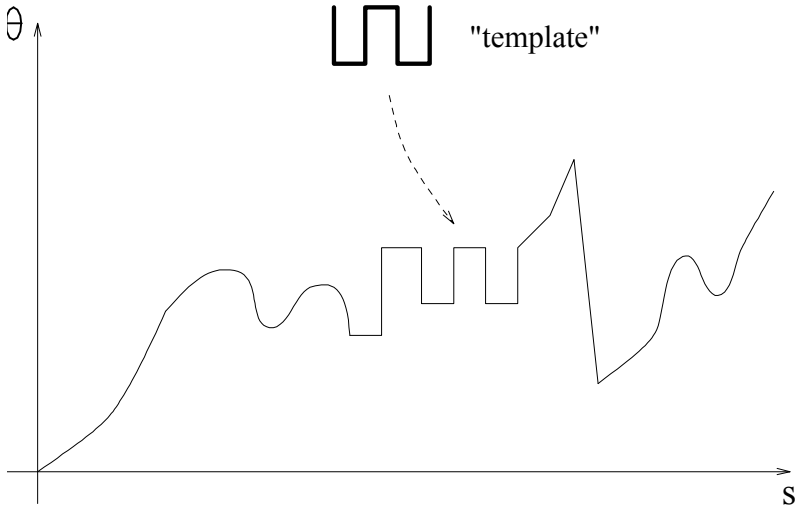


Figure 3: "template-matching" en $\theta - s$ (Turney)

La méthode proposée par Rives [*Rives*85] est assez voisine. Il utilise une approximation polygonale du contour à la place de la représentation $\theta - s$.

2 Description de la méthode

2.1 Généralités

Les méthodes décrites précédemment réalisent 2 tâches:

1. Reconnaître les objets.
2. Localiser les objets.

Ces 2 tâches sont intimement liées car ces méthodes sont basées sur des mises en correspondance avec des modèles. Il ne semble pas intéressant d'utiliser un réseau de neurones pour la seconde tâche. En effet, la localisation sert normalement à commander un robot qui va saisir l'objet, et une précision importante est requise. L'obtention d'une indication de position précise (par position on entend translation, rotation et éventuellement facteur d'échelle) en sortie d'un réseau de neurones

nécessiterait un apprentissage extrèmement long, sur un grand nombre d'exemples de diverses orientations. De plus, lorsqu'il s'agit de déterminer précisément une position, rien ne vaut le calcul géométrique, tel qu'il est par exemple mis en œuvre dans [*Ayache*86].

Par contre, l'emploi d'un réseau de neurones semble tout à fait indiqué pour une tâche de reconnaissance, car il s'agit alors d'un problème de classification. Par rapport à une méthode classique, on peut attendre d'un système à base de réseau de neurones les avantages suivants:

1. **Flexibilité**: Si de nouvelles pièces doivent apparaître sur la chaîne de fabrication, il suffit de refaire un apprentissage.
2. **Simplicité de mise en œuvre**: Lorsque le système est implanté sur un site industriel, l'opérateur n'est pas supposé être un spécialiste du traitement d'images. Il ne doit donc pas y avoir de paramètres à régler: l'ensemble doit être entièrement automatique. Cet objectif est atteint grâce aux méthodes de réglage automatique des paramètres que nous avons proposées au chapitre 3.
3. **Rapidité de reconnaissance**: en phase de reconnaissance, il suffit d'une propagation dans le réseau pour obtenir une décision.
4. **Implémentation matérielle aisée**: Du fait de sa structure répétitive et massivement parallèle , un réseau de neurones peut facilement être réalisé en logique cablée. Toutefois, il est préférable de réaliser l'apprentissage à part sur un micro-ordinateur, car il met en jeu des algorithmes relativement complexes. L'apprentissage n'a lieu normalement que lorsque des changements interviennent dans la chaîne de fabrication. En fin d'apprentissage, il suffit de charger les poids trouvés pour les connexions sur la carte matérielle.

Nous avons étudié une méthode neuronale pour la reconnaissance d'objets 2D qui présente ces avantages. Aucune modélisation n'est nécessaire car c'est un système fonctionnant par apprentissage. De plus, une faible puissance de calcul est demandée (sauf pendant la phase d'apprentissage), et la reconnaissance est très rapide.

Cette méthode permet de reconnaître les objets présents sur la scène, mais ne fournit en aucun cas leur position. L'application typique est la mesure de la proportion de chaque type de pièce avec déclenchement d'une alarme lorsque cette proportion s'écarte trop de la norme. On peut ainsi détecter des défauts tels qu'une légère déformation de l'objet perpendiculairement à son plan (fig 4), qui sont très difficilement détectables par un autre moyen. En effet, un objet déformé tombera plus souvent coté "pile" que coté "face" (ou l'inverse). Cette méthode

peut également être utilisée en association avec une méthode classique. Comme elle permet déterminer la nature des objets, ceci réduit le nombre d'hypothèses à envisager, et accélère donc le processus.

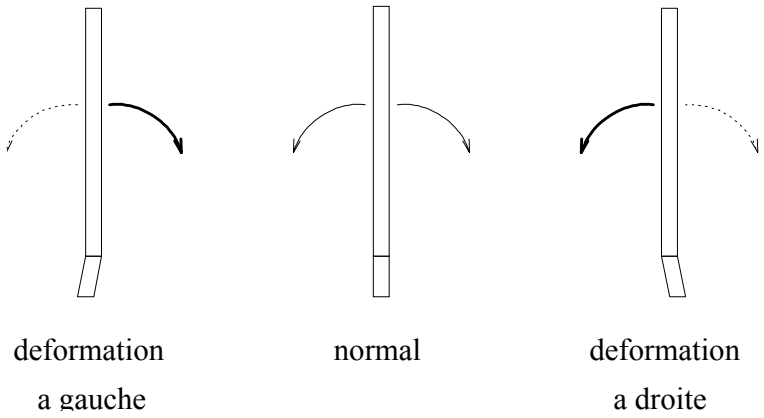


Figure 4: Exemple de défaut détectable par comptage

Une vue d'ensemble de notre méthode est fournie figure 5. La stratégie adoptée est la suivante: à partir d'un contour de l'objet à reconnaître, une aproximation polygonale est réalisée. Puis, des fonctions de reconnaissance sont calculées sur le résultat de l'approximation polygonale (typiquement, une fonction de reconnaissance est une fonction logique qui est vraie lorsqu'une configuration angulaire spécifique est trouvée sur le polygone). Les valeurs des fonctions de reconnaissance alimentent un perceptron multicouches. Pendant la phase d'apprentissage, le réseau de neurones est entrainé à reconnaître des objets partiellement occultés (les configurations d'occlusion sont choisies aléatoirement). Lorsque l'apprentissage est terminé, le système est capable de traiter de nouvelles scènes.

2.2 L'approximation polygonale

Le premier traitement à réaliser sur l'image doit avoir pour but de réduire autant que possible la quantité de données à manipuler. Cette réduction doit se faire sans perte d'information pertinente pour notre application. Une approximation polygonale des contours des objets semble être le traitement répondant au mieux à ces critères. En effet, la réduction du volume de données est considérable, car chaque segment est entièrement défini par la seule connaissance des coordonnées de ses 2 extrémités. De plus, il n'y a pas de perte d'information significative car le polygone décrit fidèlement la forme de l'objet (d'autant plus que les pièces industrielles sont souvent

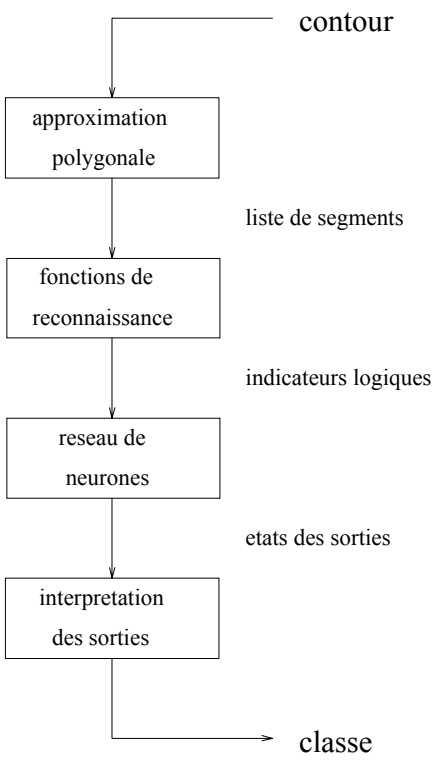


Figure 5: Méthode proposée

quasi-polygonales).

L'extraction de contour peut se faire par seuillage puis détection de zones connexes et suivi du contour des zones connexes. C'est la méthode que nous avons utilisée car les objets étaient plus clairs ou plus sombres que le fond. Si des variations d'éclairage existent sur l'image, ou si les ombres et les reflets sont importants, on peut utiliser un détecteur de contour classique (le détecteur de Deriche [*Deriche*87] semble bien indiqué car il nécessite peu d'opérations).

Il existe des techniques sophistiquées pour réaliser l'approximation polygonale d'un contour [*Leu*88]. Toutefois, nous avons utilisé une technique simple car la présence d'un réseau de neurones accroît la robustesse du système par rapport à un prétraitement peu performant. Il s'agit d'un algorithme itératif. Par exemple supposons que l'approximation courante soit celle de la figure 6a. Pour chaque segment, on identifie le point du contour associé à ce segment qui en est le plus éloigné. Si la distance entre ce point et le segment est supérieure à un seuil, on utilise ce point pour casser le segment en deux nouveaux segments, comme indiqué sur la figure 6b. Ce processus est itéré jusquà ce qu'il n'y ait plus de changement.

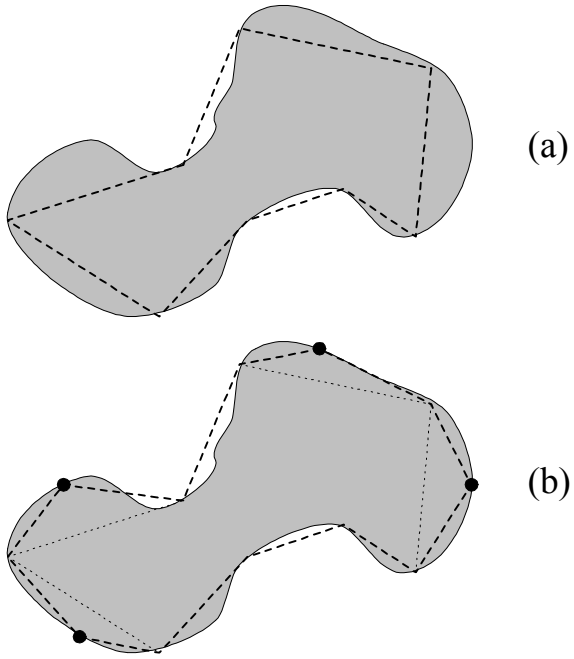


Figure 6: Approximation polygonale

La robustesse de cet algorithme est faible: le polygone obtenu dépend du point de départ sur le contour, et est fortement affecté par le bruit. Mais il semble que le réseau de neurones ait été capable d'apprendre à compenser les défauts de l'approximation polygonale car les taux de reconnaissance obtenus sont élevés, comme nous le verrons plus loin.

2.3 Les fonctions de reconnaissance

Afin de faciliter la tâche du réseau de neurones, il est souhaitable de compléter le prétraitement dans le but d'introduire des invariances en translation, rotation et facteur d'échelle. Il serait en effet sans intérêt de forcer le réseau à apprendre ces invariances. Si l'on décrit une suite de segments par leurs angles relatifs et leur longueurs, on ne perd pas l'information de forme (disposition relative des segments), mais on obtient une description invariante en translation et rotation. L'information de longueur des segments est assez peu fiable car un segment du modèle peut assez facilement être cassé en 2 segments dans la scène.

Nous sommes allés plus loin, en ne nous intéressant qu'aux configurations angulaires. On perd certes de l'information dans ce processus, mais on gagne en robustesse, et

on obtient une description invariante en translation, rotation et facteur d'échelle.

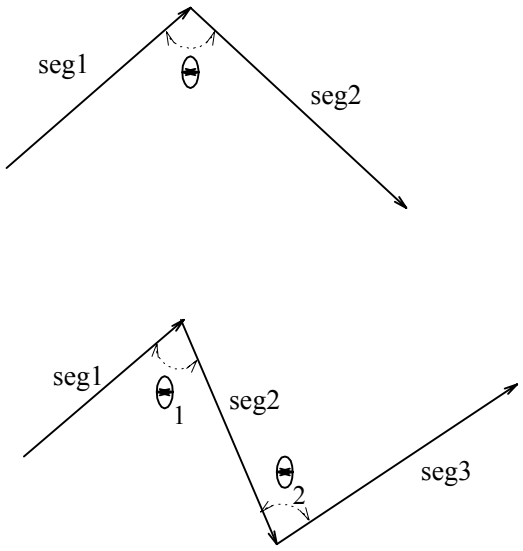


Figure 7: Fonctions de reconnaissance

Nous définissons une fonction de reconnaissance comme une fonction logique qui vaut VRAI (+1) quand une certaine configuration angulaire de segments est trouvée sur le polygone, et FAUX (-1) sinon. Nous avons utilisé des fonctions correspondant à 2 et 3 segments successifs, comme illustré sur la figure 7. Elles sont définies ci-dessous:

Pour 2 segments successifs:

$F_\theta(polygone) = VRAI$ si il existe quelque part sur le polygone 2 segments successifs dont l'angle relatif est θ (avec une certaine tolérance, par exemple 7^o).

$=$ FAUX sinon.

Pour 3 segments successifs:

$F_{\theta_1,\theta_2}(polygone) = VRAI$ si il existe sur le polygone 3 segments successifs seg1,seg2,seg3 dont l'angle relatif est θ_1 pour (seg1,seg2), et θ_2 pour (seg2,seg3) (avec une certaine tolérance, par exemple 13^o).

$=$ FAUX sinon.

Pour les fonctions correspondant à 2 segments, nous calculons toutes les fonctions avec un pas de 10^o: $F_0, F_{10}, F_{20}, ..., F_{350}$ (36 fonctions sont donc calculées).

Pour les fonctions correspondant à 3 segments, seulement les configurations (couples d'angles entre 3 segments successifs) trouvées sur des objets prototypes sont prises en compte (pour les objets présents sur la figure 9, le système a trouvé 75 configurations).

Le nombre total de fonctions de reconnaissance adaptées à l'application présentée dans le paragraphe 5 est donc 111. Ce nombre fixe la taille de la couche d'entrée du réseau de neurones.

2.4 Le réseau de neurones

Nous avons utilisé un réseau à 3 couches, entraîné par l'algorithme de rétropropagation, dans sa version améliorée (chap 3). Le réseau est entièrement connecté entre 2 couches successives et est seuillé. La fonction d'activation du neurone est la tangente hyperbolique (cf notations du chapitre 2):

$$O_j = F(X_j) = th(X_j)$$

La couche d'entrée est alimentée avec les valeurs des fonctions de reconnaissance. Pour un problème à 2 classes, comme ce fut le cas pour nos expérimentations, il y a 2 neurones en sortie (un par classe), et les sorties désirées sont:

$$(+1, -1) \text{ pour la classe 1}$$
$$(-1, +1) \text{ pour la classe 2}$$

L'apprentissage se fait conformément à l'algorithme décrit au chapitre 2 avec les améliorations du chapitre 3. Lorsque l'apprentissage est terminé, le réseau peut traiter de nouvelles données. C'est le neurone dont la sortie est la plus forte qui détermine la classe.

Une mesure de confiance (entre 0 et 1) peut aussi être définie:

$$\text{confiance} = 0.5 \mid \text{plus forte sortie} - \text{sortie suivante} \mid$$

3 Un système d'apprentissage symbolique: AGRRAPE

[Cet algorithme a été développé aux LER pour une autre application]

3.1 Introduction

Dans le cadre de la construction de systèmes experts, nombre de méthodes pour induire des règles symboliques à partir d'exemples ont été proposées. Les principales familles de méthodes sont basées sur les algorithmes ID3 [*Quinlan*86] et AQ [*Kononenko*84].

L'algorithme ID3 produit ses sorties sous forme d'un arbre de décision, qui est généralement difficilement compréhensible et difficile à manipuler. De plus, l'application de ID3 dans un domaine réel nécessite un traitement complexe pour supporter des données bruitées.

L'algorithme AQ génère un ensemble de règles de classification qui sont compréhensibles et faciles à manipuler. Malheureusement, les règles générées dépendent de l'ordre de présentation des exemples, et l'application à des données bruitées nécessite la mise en oeuvre de post-traitements complexes.

Nous avons développé un nouvel algorithme d'induction [*Burel*90], AGRRAPE (Algorithme de Génération de Règles de Reconnaissance A Partir d'Exemples). Les principales caractéristiques d'AGRRAPE sont les suivantes:

- Ses sorties sont des règles compréhensibles et faciles à manipuler.
- Il peut facilement traiter des données bruitées sans nécessiter de post-traitement complexe.
- Contrairement à ID3, il ne défavorise pas les classes représentées par peu d'exemples

Pour présenter AGRRAPE, nous utilisons un ensemble d'apprentissage limité, représenté dans le tableau suivant (cet ensemble est sans rapport avec notre application et a simplement été constitué dans un but d'illustration).

Objet	*Attributs*					*Classe*
	A	B	C	D	E	
O_1	a_1	b_1	c_1	d_1	128.5	C_1
O_2	a_2	b_1	c_2	d_1	228.5	C_1
O_3	a_2	b_1	c_2	d_3	12	C_1
O_4	a_5	b_1	c_2	d_3	124	C_1
O_5	a_1	b_1	c_1	d_1	45	C_1
O_6	a_4	b_3	c_2	d_2	36.75	C_2
O_7	a_2	b_3	c_1	d_2	18.8	C_2
O_8	a_1	b_1	c_1	d_2	128	C_2
O_9	a_3	b_3	c_2	d_1	14.9	C_2
O_{10}	a_3	b_2	c_2	d_1	228.5	C_3
O_{11}	a_4	b_2	c_2	d_1	35.9	C_3
O_{12}	a_3	b_3	c_1	d_3	45.78	C_3
O_{13}	a_5	b_3	c_1	d_1	102	C_3
O_{14}	a_2	b_3	c_2	d_2	105	C_3

Chaque objet $O_1, ..., O_{14}$ appartient à l'une des classes C_1, C_2, C_3 et est décrit par 5 attributs:

- Trois attributs symboliques non ordonnés
  - A à valeur dans a_1, a_2, a_3, a_4, a_5
  - B à valeur dans b_1, b_2, b_3
  - C à valeur dans c_1, c_2
- Un attribut ordonné
  - D à valeur dans $d_1,\ d_2,\ d_3$ avec:

$$d_1 < d_2 < d_3$$

- Un attribut numérique E

3.2 La représentation des règles

AGRRAPE induit des règles de la forme:
' SI $<$ *condition* $>$ ALORS prédire $<$ *classe* $>$ avec une probabilité $< p >$ et une

généralité $< g >$'.

$<$ *condition* $>$ est une conjonction de tests d'attributs. Par exemple, (A=a_2) et (E > 20) sont des tests possibles sur les attributs A et E. AGRRAPE permet des tests avec les opérateurs $\{<, >, =, \neq\}$.
$< p >$ est la probabilité pour qu'un élément appartienne à $<$ *classe* $>$ sachant que $<$ *condition* $>$ est vérifiée.
$< g >$ est la probabilité pour qu'un élément soit couvert par la règle sachant qu'il appartient à $<$ *classe* $>$.

Ces deux probabilités sont calculées sur l'ensemble d'apprentissage.

Les règles générées à partir de l'ensemble d'apprentissage simplifié sont:

R_1: si $B = b_1$ alors C_1 avec la probabilité 0.83 et la généralité 1.0
R_2: si $B \neq b_1$ et $E > 41.5$ alors C_3 avec la probabilité 1.0 et la généralité 0.8
R_3: si $E < 41$ et $B = b_3$ alors C_2 avec la probabilité 1.0 et la généralité 0.75
R_4: si $E > 126$ et $E < 128.25$ alors C_2 avec la probabilité 1.0 et la généralité 0.25
R_5: si $B = b_2$ alors C_3 avec la probabilité 1.0 et la généralité 0.2

3.3 L'apprentissage

Nous ne décrirons pas l'algorithme d'apprentissage d'AGRRAPE, qui est détaillé dans [*Burel*90]. Disons simplement qu'il s'agit d'un algorithme itératif, qui génère des règles par spécialisation et utilise le critère de qualité suivant pour conserver uniquement les meilleures règles:

$$Q_R = P(C/R)^2 \ * \ P(R/C) \ * \ (1 - P(R/notC))$$

Où C est la classe, R la règle, et P(.) la probabilité.
Chaque règle vise à séparer au mieux une classe des autres classes. Dans notre exemple, les trois premières règles générées ont été:

règle pour la classe C_1: R_1 (cf §3.2)
règle pour la classe C_2: R_3
règle pour la classe C_3: R_2

avec les qualités:

$$Q_{R_1} = 0.6, \ Q_{R_2} = 0.75, \ Q_{R_3} = 0.9$$

3.4 La phase de reconnaissance

La reconnaissance consiste à associer une classe à un objet. Ceci est réalisé en recherchant une règle qui soit vérifiée par les attributs de l'objet. Le système teste les règles par ordre de probabilité décroissante. Trois cas sont alors possibles:

- Aucune règle n'est vérifiée: l'objet n'est pas affecté.
- Une seule règle est vérifiée: l'objet est affecté à la classe prédite par cette règle.
- Plusieurs règles de même probabilité sont vérifiées. Le système sélectionne celle qui a la plus forte généralité, et affecte l'objet à la classe prédite par cette règle.

4 Classifieur de Bayes

Dans un souci de comparaison, nous avons également développé un classifieur Bayésien adapté au problème. Nous utilisons les notations du chapitre 2.
La fonction de coût choisie est:

$$\begin{aligned} C(\theta,\delta) &= 0 \ si \ \delta = \theta \\ &= 1 \ sinon \end{aligned}$$

C'est à dire que l'on ne privilégie aucune classe. On a vu que le risque moyen est minimisé si on choisit la décision δ qui minimise

$$\sum_{\theta} C(\theta,\delta) P(X,\theta)$$

c'est à dire, compte tenu du choix de $C(\theta,\delta)$:

$$\sum_{\theta \neq \delta} P(X,\theta)$$

Or, comme $\sum_{\theta \neq \delta} P(X,\theta) = P(X) - P(X,\delta)$, ceci revient à maximiser $P(X,\delta)$. Finalement, on a donc à prendre la décision $\delta = \theta$ qui maximise $P(X|\theta)P(\theta)$.

Les $P(\theta)$ peuvent être estimés sur l'ensemble d'apprentissage. Pour les $P(X|\theta)$, une approximation est nécessaire car il y a 2^{111} valeurs de X possibles. On suppose les fonctions de reconnaissance indépendantes (ce qui n'est certainement pas tout à fait exact, mais ce type d'approximation est souvent nécessaire pour mettre en oeuvre un classifieur Bayésien dans une application réelle), et on estime sur l'ensemble d'apprentissage les $P(X_i|\theta)$, où X_i est une fonction de reconnaissance. En exploitant l'hypothèse d'indépendance, on peut écrire:

$$P(X|\theta) = \prod_i P(X_i|\theta)$$

5 Résultats expérimentaux

5.1 Conditions expérimentales

Les expérimentations ont été conduites sur un problème de "pile ou face". La figure 9 montre le type de pièce traitée. Il s'agit de pièces de magnétophone. Il y a deux classes car ces pièces ont pu tomber coté "pile" ou coté "face".

Il aurait été trop coûteux d'acquérir des centaines d'images d'objets partiellement occultés afin d'entrainer et de tester notre système. Aussi, la validation de notre méthode a été réalisée sur une base d'images synthétisées à partir de l'image représentée figure 9. Un seuillage, suivi d'une détection de zones connexes permet d'affecter un label à chaque objet de cette image (le résultat est représenté figure 10). Puis, une base de données de 100 images (telle que l'image de la figure 11) avec de l'ordre de 6 morceaux d'objets par image a été générée par translation et rotation aléatoire des objets de la figure 10. Les morceaux d'objets dont la surface est inférieure à 20% de la surface totale de l'objet ont été supprimés, car ils sont inclassifiables, même pour l'observateur humain. 79 images ont été utilisées pour l'apprentissage (elles contenaient au total 481 morceaux d'objets) , et les 21 images restantes ont été utilisées pour l'évaluation (elles contenaient un total de 126 morceaux d'objets).

5.2 Comparaisons

Le réseau de neurones était entièrement connecté entre couches successives, et seuillé. Il comprenait 3 couches, avec un nombre de neurones par couche de 111, 4 et 2

(neurones seuils non compris).

Le taux de reconnaissance obtenu sur l'ensemble d'évaluation est de 60/64 pour la classe 1, et de 61/62 pour la classe 2. Le taux global de reconnaissance est donc 121/126 = **96%**. Les figures 12 et 14 représentent les résultats de la classification (un niveau de gris par classe) des figures 11 et 13

Une comparaison avec un système d'apprentissage symbolique (AGRRAPE) alimenté avec les mêmes données a été réalisé. Nous avons utilisé les mêmes ensembles d'apprentissage et d'évaluation, et le même prétraitement (approximation polygonale et fonctions de reconnaissance). Le taux de reconnaissance obtenu sur l'ensemble d'évaluation est de 54/64 pour la classe 1 et de 52/62 pour la classe 2. Le taux global de reconnaissance est donc 106/126 = **84%**. 15/126 = **12%** des morceaux d'objets ont été mal classifiés et **4%** n'ont pas été assignés. Le système symbolique a généré 26 règles, dont la partie condition contenait 3 tests d'attributs en moyenne. Afin de comparer les résultats en classification forcée, on peut supposer que pour les 4% non assignés, on tire la classe au hasard. Le taux de reconnaissance est alors de 86%.

En dépit d'un taux de reconnaissance plus faible (86% contre 96%), le principal intérêt d'un système symbolique est le fait que le processus utilisé pour classifier peut être compris, et il est possible de modifier les règles générées pour améliorer les performances, lorsque l'on connait bien l'application. Ce système symbolique a été initialement conçu pour traiter des problèmes dans lesquels on dispose de données hiérarchiques complexes. Le réseau de neurones est plus rapide en apprentissage mais par contre plus lent en reconnaissance, car la mise en oeuvre des règles générées par AGRAPPE ne nécessite que des opérations de comparaison.

Le classifieur Bayésien (§4), ainsi qu'un classifieur aux 3 plus proches voisins (chap 2) ont également été utilisés dans les mêmes conditions. Les taux de reconnaissance sont de 42/64 et 60/62, soit 81% pour le classifieur Bayésien, et de 56/64 et 56/62 soit 89% pour le classifieur aux 3 plus proches voisins (c'est pour ce nombre de voisins que l'on obtient le meilleur résultat). Le tableau suivant indique les taux de reconnaissance et le nombre de multiplications requis:

classifieur	taux de reconnaissance	nombre de multiplications
RN	96%	452
AGRRAPE	86%	0
Bayésien	81%	222
3-ppv	89%	53391

Afin de vérifier la généralité des taux de reconnaissances fournis ci dessus, 10 autres combinaisons ensemble d'apprentissage/ensemble dévaluation ont été

créées aléatoirement. Pour chaque combinaison, et pour chaque classifieur (sauf pour AGRRAPE), on calcule le taux de reconnaissance obtenu sur l'ensemble d'évaluation. Le taux de reconnaissance moyen (par classifieur) ne s'éloigne pas de plus de 1.5% des taux donnés ci-dessus, et l'écart type sur le taux de reconnaissance ne dépasse pas 2.5%. Les résultats fournis dans le tableau ci dessus sont donc significatifs.

5.3 Autres résultats

Afin de prouver la validité de l'algorithme sur des images qui ne sont pas des images de synthèse, nous présentons le résultat du traitement de quelques images réelles (figures 15 à 17) avec le réseau précédent. Le facteur d'échelle est inconnu. Après extraction des contours de l'image et approximation polygonale, on classifie chaque segment (gris pour la classe 0, blanc pour la classe 1) de la façon suivante:

On considère le groupe de 5 segments qui entoure le segment considéré (fig 8). On calcule les fonctions de reconnaissance sur ce groupe, et on propage dans le réseau de neurones. Si la confiance obtenue est suffisante (supérieure à 0.5), on affecte au segment central la classe trouvée. Sinon, on rajoute 2 segments de chaque coté, et ainsi de suite jusquà ce que la confiance dépasse 0.5, ou que l'on ait atteint 13 segments. Dans ce dernier cas, on décidera de la classe qui avait fourni la plus forte confiance.

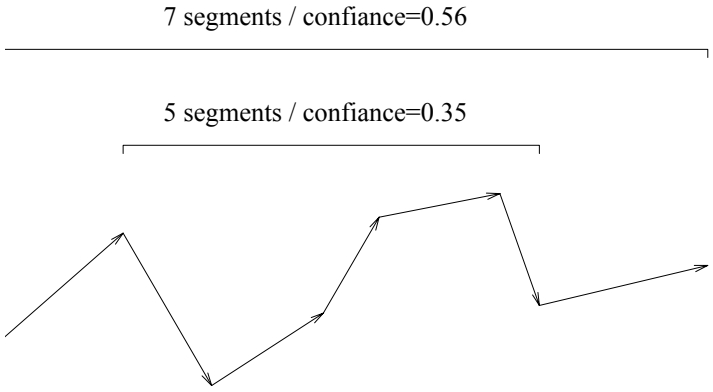


Figure 8: Evolution du nombre de segments pris en compte en classification locale

Les figures 15 à 17 montrent que la classification obtenue est généralement correcte, mis à part quelques erreurs isolées. Pour l'application de détection d'un défaut de la chaîne de fabrication qui provoque des déformations des objets dans leur plan, on

somme, pour chaque classe et sur plusieurs images, la longueur des segments visibles affectés à cette classe. Si les objets ne sont pas déformés, on doit trouver des sommes sensiblement égales.

6 Conclusion

Nous avons décrit un système connexionniste pour la reconnaissance d'objets partiellement visibles, qui a l'avantage d'être facile à implémenter et à mettre en œuvre. Aucune modélisation n'est nécessaire car il fonctionne par apprentissage. De plus, une faible puissance de calcul est demandée (sauf pendant la phase d'apprentissage), et la reconnaissance est très rapide. Le temps de calcul requis est de quelques secondes par image, prétraitement compris, sur une station de travail de type SUN4 (programme écrit en C), à comparer par exemple avec les 20mn par objet annoncés par Turney [*Turney*84] sur VAX11/780.
Cette méthode ne localise pas les objets. Elle peut être utilisée seule (pour un comptage par exemple) ou en association avec un système classique (dans ce cas, elle devrait accélérer la reconnaissance en guidant la génération d'hypothèses).

Une comparaison avec un système symbolique a également été réalisée. Elle montre les avantages et les inconvénients de chaque méthode. Le choix entre les 2 systèmes dépend de l'importance relative accordée à des paramètres tels que le taux de reconnaissance, la vitesse, l'adaptabilité à d'autres prétraitements (générant éventuellement des données hiérarchiques), la possibilité d'agir sur le système pour y inclure des connaissances, la puissance de calcul requise, la complexité d'implémentation, la possibilité d'examiner le système quand l'apprentissage échoue,...

Les résultats obtenus semblent satisfaisants. Un travail futur pourrait comprendre l'amélioration du prétraitement, et la présentation d'une information plus complète au réseau de neurones. Cette information pourrait par exemple inclure la longueur des segments, mais on perd alors l'invariance en facteur d'échelle.

Références

[*Ayache*86] N.AYACHE, O.D.FAUGERAS
"HYPER: A new approach for the recognition and positioning of 2D objects"
IEEE-PAMI, vol 8, $n°1$, January 1986

[*Bolles*82] R.C.BOLLES, R.A.CAIN
"Recognizing and Locating Partially Visible Objects:
The Local-Feature-Focus Method"
The International Journal of Robotics Research
vol 1, $n°3$, Fall 1982

[*Burel*90] G. BUREL, D. CAREL, J.Y. CATROS
"A connectionist system for recognition of 2D workpieces"
Revue Technique THOMSON-CSF *vol* 22, n^o 4, décembre 1990

[*Deriche*87] R. DERICHE
"Using Canny's Criteria to Derive a Recursively Implemented
Optimal Edge Detector"
International Journal of Computer Vision, pp 167-187, 1987

[*Grimson*87] W.E.L.GRIMSON, T.LOZANO-PEREZ
"Localizing overlapping parts by searching the interpretation tree"
IEEE-PAMI, vol 9, $n°4$, July 1987

[*Koch*87] M.W.KOCH, R.L.KASHYAP
"Using polygons to recognize and locate partially occluded parts"
IEEE-PAMI, vol 8, $n°4$, July 1987

[*Kononenko*84] I.KONONENKO, I.BRATKO,E.ROSKAR
"Experiments in automatic learning of medical diagnostic rules"
Technical Report: E.Kardelj University, Faculty of Electrical
Engineering, 1984.

[*Leu*88] J.G.LEU, L.CHEN
"Polygonal approximation of 2D shapes through boundary merging"
Pattern Recognition Letters 7, pp231-238, April 1988

[*Michalsky*69] R.S.MICHALSKY
"On the quasi-minimal solution of the general covering problem"
Proceeding of the Fifth International Symposium on Information Processing
pp125-128, 1969.

[*Quinlan*86] J.R.QUINLAN
"Induction of decision trees"
Machine learning 1, pp81-106, 1986

[*Rives*85] G. RIVES, M. DHOME, J.T. LAPRESTE, M. RICHETIN
"Detection of patterns from piecewise linear contour"
Pattern Recognition Letters 3, March 1985, pp99-104

[*Rumelhart*86] D.E. RUMELHART, G.E. HINTON, R.J. WILLIAMS
"Learning internal representations by error backpropagation"
Parallel Distributed Processing, D.E. RUMELHART and J.L. Mc CLELLAND
Chap 8, Bradford Book - MIT Press - 1986

[*Simon*88] S.SIMON, J.P.ROGALA
"Model-based prediction-verification scheme for real-time inspection"
Pattern Recognition Letters 7, pp305-311, June 1988

[*Turney*84] J.L.TURNEY, T.N.MUDGE, R.A.VOLZ
"Recognizing partially occluded parts"
IEEE-PAMI, vol 7, $n°4$, July 1984

Figure 9: Image source

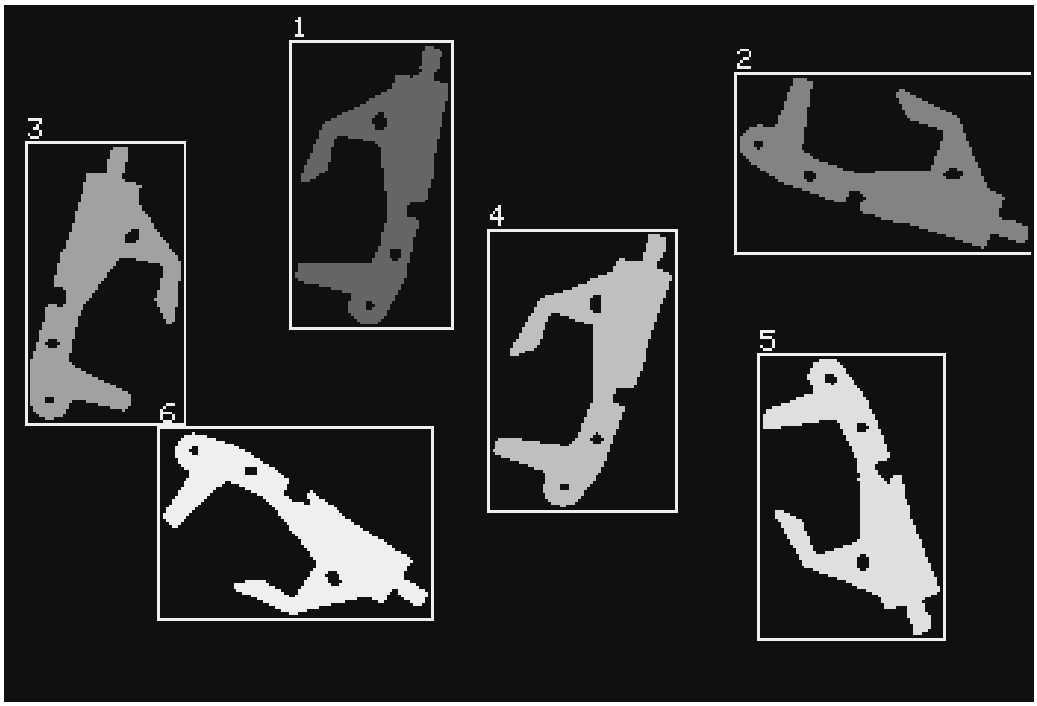


Figure 10: Recherche des objets

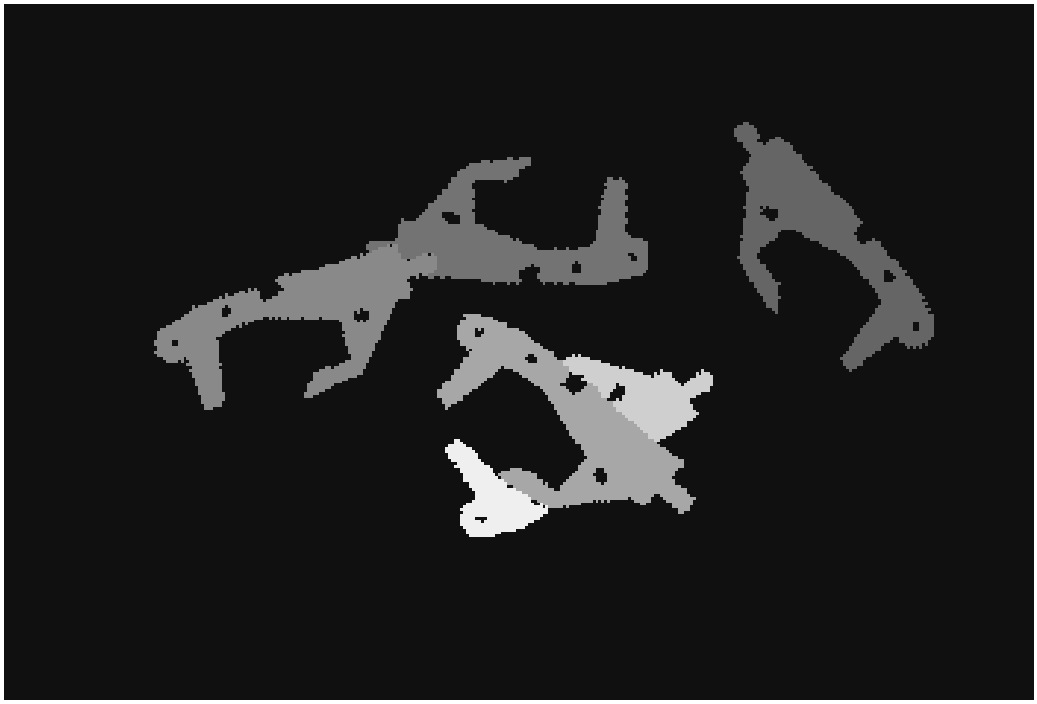


Figure 11: Image à classifier

Figure 12: Classification de l'image précédente

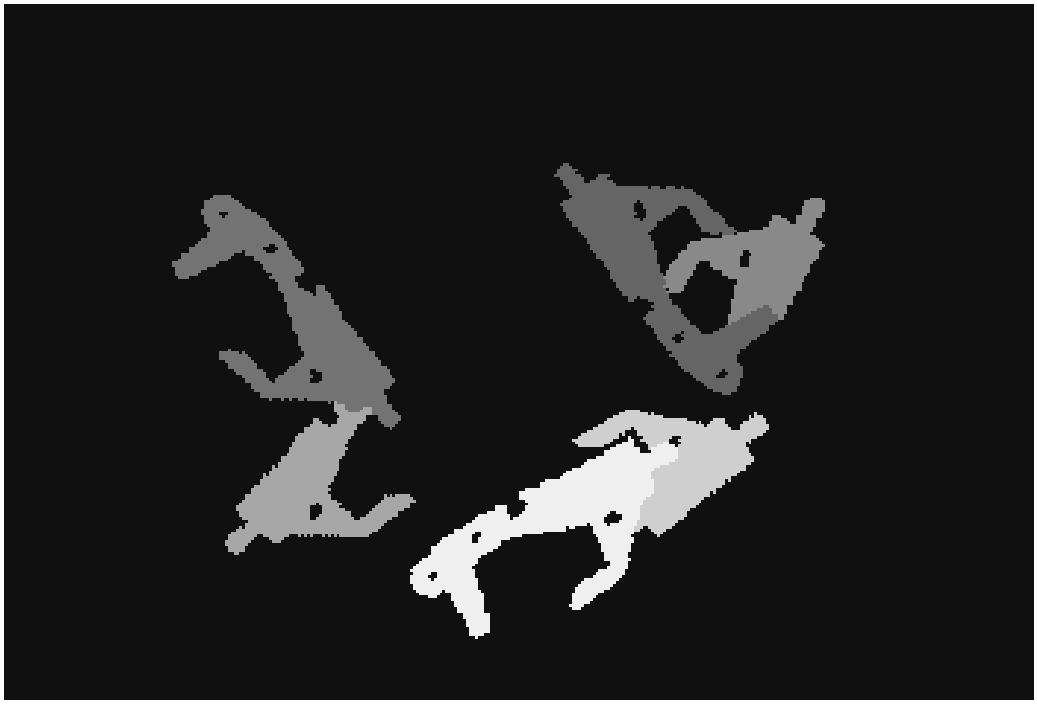


Figure 13: Image à classifier

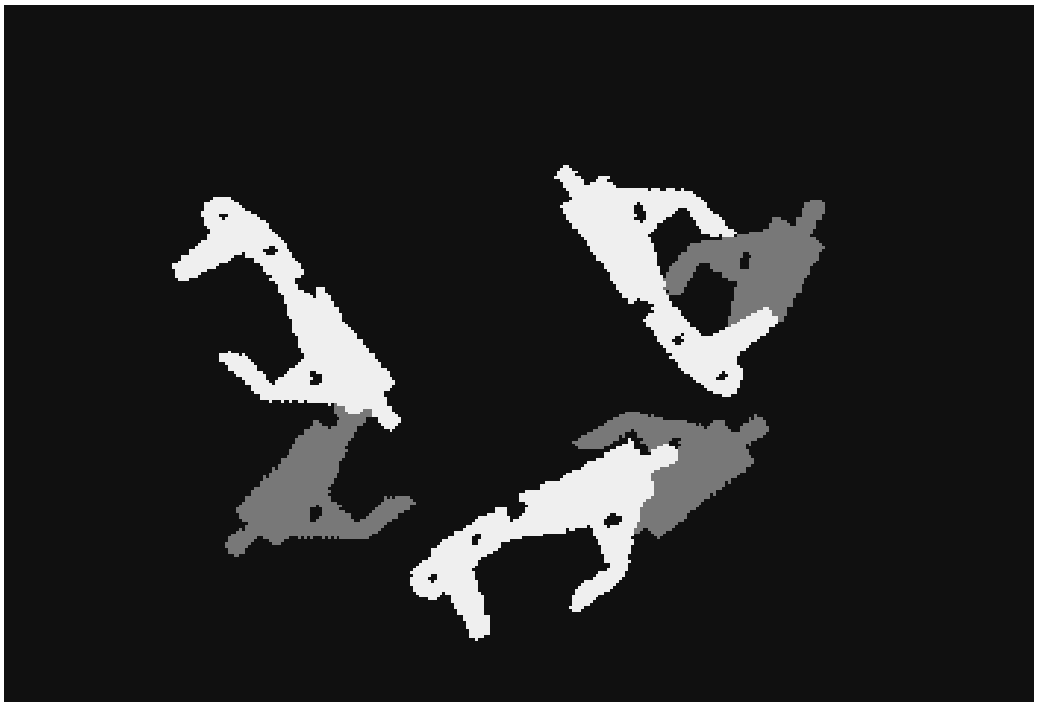


Figure 14: Classification de l'image précédente

Figure 15: Classification des contours d'une image réelle

Figure 16: Classification des contours d'une image réelle

Figure 17: Classification des contours d'une image réelle

Chapitre 7:

RECONNAISSANCE D'OBJETS 3D

De nombreuses applications de vision, notamment en robotique, nécessitent l'identification d'objets tri-dimensionnels. L'emploi de méthodes complexes, à base de mise en correspondance avec des modèles, n'est pas toujours justifié, et peut être jugé trop coûteux en temps de calcul. Nous proposons une méthode rapide, et de mise en œuvre aisée, pour la reconnaissance d'objets 3D.
La méthode exploite les capacités d'apprentissage et de généralisation d'un perceptron multicouches. L'idée consiste à faire apprendre à un tel réseau un certain nombre de vues de chaque objet possible. Pour réduire la quantité de données à traiter, l'objet est caractérisé par sa silhouette. En fin d'apprentissage, les capacités de généralisation du perceptron multicouches lui permettent une reconnaissance sur des vues non apprises.

Après une description de la méthode proposée, nous présenterons les résultats obtenus sur une base de données de 216 images.

1 Introduction

La reconnaissance automatique d'objets tri-dimensionnels est considérée comme une tâche difficile du fait de l'importance des données à traiter et de la complexité de l'information sous-jacente. En raison de l'essort de la robotique, ce domaine est appelé à se développer dans l'avenir.

Dans la littérature, plusieurs approches ont été proposées pour la reconnaissance d'objets 3D . La plupart des méthodes sont basées sur la comparaison avec des modèles des objets à reconnaître [$DeGreef89$]. On peut en première approximation les classer en 2 grandes catégories:

- Méthodes exploitant plusieurs vues de l'objet.
- Méthodes exploitant une seule vue de l'objet.

Le premier type de méthode consiste à reconstruire en 3 dimensions l'objet présent dans la scène (en mettant en œuvre des techniques de type stéréoscopie), puis à réaliser la mise en correspondance avec des modèles 3D. Ces méthodes supposent la mise en place d'un dispositif d'acquisition important (au moins 2 caméras, parfaitement calibrées), et nécessitent un traitement informatique complexe (pour résoudre le problème de la mise en correspondance). Mais elles présentent l'intérêt de fournir la position précise de l'objet en plus de la reconnaissance.

Le second type de méthode consiste à tenter d'interpréter l'image comme la projection de l'un des modèles 3D dont on dispose. La focalisation sur des structures caractéristiques (par exemple des groupes de 3 segments adjacents) permet de réduire la complexité du problème. Pour une illustration de ce type de méthode, on pourra consulter [$DeGreef90$].

Les 2 types de méthodes nécessitent la création préalable de modèles des objets. Ceci explique probablement le fait que les solutions proposées sont généralement limitées aux objets polygonaux [$DeGreef89$] [$Guerra90$] . Les modèles polygonaux conduisent à manipuler des segments, alors que des modèles plus généraux nécessiteraient la gestion de primitives complexes. De plus, les segments sont des primitives relativement facile à détecter dans une image.

Du fait de leur capacité d'apprentissage, les réseaux de neurones pourraient permettre d'éviter la phase de modélisation. Même si les performances obtenues (taux de reconnaissance) sont finalement plus faibles que celles des systèmes à base de modèle, ceci est une propriété intéressante qui permettrait de réduire le coût de mise au point d'un système. L'idée consiste à apprendre à un réseau de neurones

un certain nombre de vues de chaque objet. En fin d'apprentissage, il devrait être capable de reconnaître des vues non apprises.

Nous avons spécifié et expérimenté une approche qui est basée sur la reconnaissance de l'objet à partir de sa silhouette (contour externe). La silhouette peut être entièrement décrite par un ensemble de coefficients complexes connus sous le nom de "descripteurs de Fourier" [*Granlund*72]. Cette description dans le domaine de fourier présente les avantages suivants:

- Les descripteurs de Fourier portent la même information que le contour original (i.e. il est possible de reconstruire le contour à partir des descripteurs de Fourier).
- Ils procurent une information multi-résolution.
- Il est possible de les normaliser pour les rendres invariants par rapport aux transformations de translation, rotation, et homothétie dans le plan, ainsi que par rapport à un décalage du point de départ sur le contour.

Le paragraphe suivant est une rapide description de la base d'images constituée pour les expérimentations et du traitement bas niveau mis en œuvre. Ensuite, nous détaillerons la méthode adoptée pour utiliser les descripteurs de Fourier en association avec un réseau de neurones pour l'identification d'objets 3D. La méthode inclut une normalisation de telle sorte qu'ils soient invariants en translation, rotation et facteur d'échelle. Il s'agit bien entendu d'invariances 2D et non 3D, c'est à dire que les coefficients normalisés sont invariants par rapport à une translation de l'objet dans le plan image, à une rotation de la caméra autour de son axe et à une modification de la distance caméra-objet. Nous étudions également diverses stratégies pour fournir les données au réseau de neurones dans une représentation adéquate. Les résultats expérimentaux en apprentissage et généralisation sont ensuite présentés.

2 La base d'images et le traitement bas niveau

2.1 La base d'images

Les expérimentations ont été conduites sur une base de 216 images dont la taille est d'environ 300x200 pixels. Il y a 3 objets différents: un distributeur de ruban adhésif (SCOTCH), une agrafeuse (STAPLER) et une pièce industrielle (WORKPIECE). Pour chaque objet, il y a deux séries de 36 images. Elles ont été obtenues en plaçant

l'objet sur un support que l'on fait tourner par pas de 10^o autour de l'axe vertical. Une série a été réalisée avec la caméra sous une élévation de 0^o, et l'autre série avec la caméra sous une élévation de 30^o. Dans les 2 cas, la caméra vise approximativement le centre de gravité de l'objet.

Ces 216 images ont été divisées en 2 groupes de 108 images: l'ensemble d'apprentissage, qui permet d'entraîner notre système, et l'ensemble d'évaluation qui permet de tester ses performances. L'ensemble d'apprentissage contient les images prises sous un azimut multiple de 20^o ($0^o, 20^o, 40^o, ..., 340^o$). L'ensemble d'évaluation contient les autres images (azimut $= 10^o, 30^o, ..., 350^o$).

Quelques images sources sont visibles sur les figures 1 à 3. On remarquera des problèmes d'ombres et de reflets. Ceci est particulièrement vrai pour la pièce industrielle, dont la surface métallique produit d'importants reflets, de sorte que les contours extraits seront très bruités. Mais comme nous le verrons plus loin, les performances obtenues en reconnaissance sont satisfaisantes, ce qui confirme la robustesse de la méthode.

2.2 Extraction de la silhouette

L'objet est en moyenne de brillance supérieure au fond. Par conséquent, un simple seuillage permet d'isoler la plus grande partie des pixels correspondant à l'objet. On a utilisé ici un seuil fixe, réglé légèrement au dessus du niveau moyen du fond. Cependant, du fait des reflets sur le fond, certaines zones du fond passent le seuil. Afin de les éliminer on effectue une détection de zones connexes. Seule la zone connexe de plus forte surface est conservée, ce qui permet d'éliminer les reflets sur le fond, et de ne conserver que l'objet. Puis, on effectue un suivi de contour.

Pour information, quelques contours d'objets sont présentés sur les figures 4 à 7. On remarquera les perturbations importantes sur les contours (principalement pour la pièce industrielle), dûes aux problèmes d'ombres et de reflets.

3 Les descripteurs de Fourier

3.1 Introduction

La théorie des descripteurs de Fourier est une méthode de décomposition d'une forme en série de Fourier [*Granlund*72] [*Mitchell*83]. Il s'agit d'une transformation qui présente des propriétés intéressantes: les opérations de translation, rotation, dilatation et permutation circulaire des indices s'expriment de façon très simple dans le domaine transformé. De nombreux auteurs ont étendu la liste des propriétés connues des descripteurs et les méthodes de normalisation [*Mitchell*83] [*Lin*87].

Etant donné le contour (dans le sens trigonométrique) d'un objet sous forme d'une liste de coordonnées de pixels, nous construisons un contour de vélocité uniforme de N points équidistants en re-échantillonnant le contour original. Sur le contour original, la distance entre deux pixels 4-connexes est 1.0 et la distance entre deux pixels 8-connexes est $\sqrt{2}$. Après re-échantillonage, toutes les distances entre points successifs sont identiques. Un re-échantillonage plus élaboré, utilisant par exemple une fine approximation polygonale pour la définition de l'abscisse curviligne, aurait pu être envisagée pour des images de plus faible résolution. Il n'est pas justifié ici.

Notons U_m les points du contour re-échantillonné (ils peuvent être vus comme des vecteurs dans le plan complexe). Les descripteurs de Fourier sont définis comme:

$$C_n = \frac{1}{N} \sum_{m=0}^{N-1} U_m e^{-j2\pi \frac{nm}{N}}$$

Il y a N descripteurs de Fourier, et les indices peuvent être vus modulo N (car $C_n = C_{n \ mod \ N}$). La transformation inverse est donnée par :

$$U_m = \sum_{n=0}^{N-1} C_n e^{j2\pi \frac{nm}{N}}$$

3.2 Normalisation

Les transformations géométriques de base (translation, rotation, homothétie) et le décalage du point de départ sur le contour agissent sur les descripteurs de Fourier comme suit:

Translation	$U'_m = U_m + T$	$C'_0 = C_0 + T$
Rotation	$U'_m = U_m e^{j\phi}$	$C'_n = C_n e^{j\phi}$
Homothétie	$U'_m = \rho U_m$	$C'_n = \rho C_n$
Décalage	$U'_m = U_{(m+k) \ mod \ N}$	$C'_n = C_n e^{2j\pi\frac{kn}{N}}$

En conséquence, pour obtenir l'invariance par rapport à ces transformations, nous réalisons les normalisations suivantes:

1. C'_0 n'est pas pris en compte, donc l'invariance en TRANSLATION est assurée.
2. Tous les coefficients sont divisés par C'_1, donc les invariances en ROTATION et HOMOTHETIE sont assurées. Ceci peut aisément être vérifié:
$$\begin{aligned} K'_n &= \frac{C'_n}{C'_1} \\ &= \frac{\rho C_n e^{j\phi} e^{2j\pi\frac{kn}{N}}}{\rho C_1 e^{j\phi} e^{2j\pi\frac{k}{N}}} \\ &= \frac{C_n}{C_1} e^{2j\pi\frac{k(n-1)}{N}} \\ &= K_n e^{2j\pi\frac{k(n-1)}{N}} \end{aligned}$$
3. Tous les coefficients K'_n sont ensuite multipliés par $e^{j(1-n)\phi'_2}$, où ϕ'_2 est l'argument de K'_2. Comme $\phi'_2 = \phi_2 + 2\pi\frac{k}{N}$, nous avons:
$$\begin{aligned} P'_n &= K'_n e^{j(1-n)\phi'_2} \\ &= K_n \left(e^{2j\pi\frac{k(n-1)}{N}}\right)\left(e^{j(1-n)(\phi_2+2\pi\frac{k}{N})}\right) \\ &= K_n e^{j(1-n)\phi_2} \\ &= P_n \end{aligned}$$
Donc l'invariance par rapport au point de départ sur le contour est assurée. Ici, il est important de noter que l'argument ϕ'_2 doit être arrondi à la valeur la plus proche de la forme $\frac{2\pi}{N}$, où l est un entier. Ne pas le faire causerait l'incohérence des descripteurs normalisés, car il ne serait plus possible de les interpréter comme étant les descripteurs d'un objet réel, obtenus en faisant subir à l'objet source des transformations géométriques de base, et un décalage du point de départ sur le contour. Ceci peut facilement être vérifié en reconstruisant le contour à partir des descripteurs de Fourier.

Cette normalisation n'est efficace que sous réserve que la norme des coefficients C_1 et C_2 ne soit pas trop faible. En effet, dans le cas contraire, l'imprécision sur leur

argument est importante. Nous montrons en annexe 1 que la norme du coefficient C_1 est le rayon du cercle qui approxime au mieux la forme. Cette norme est donc assez élevée pour les objets réels. Quant à la norme de C_2, la figure 9 (sur laquelle nous reviendrons), montre quelle est généralement assez importante. Lorsque ceci n'est pas le cas, on peut envisager une méthode de normalisation plus élaborée.

Il est important de noter qu'il n'y a pas de perte d'information conçernant la forme des objets dans notre prétraitement. L'application de la transformation inverse aux coefficients normalisés P_n fournit une silhouette identique à l'originale, à une transformation géométrique près (composition d'une translation, d'une rotation, et d'une homothétie).

La figure 8 montre la reconstruction de l'agrafeuse (vue sous 0^o d'élévation et 90^o d'azimut) en fonction du nombre de coefficients conservés. On ne conserve que les coefficients d'indice n tel que $-\Delta \leq n \leq \Delta$, pour les valeurs $\Delta = 1, 2, 4, 5, 6, 8, 10, 16, 32$. On constatera que les coefficients d'ordre inférieur à 32 sont très largement suffisants pour reconstruire fidèlement la forme. Les coefficients d'ordre élevé correspondent à des hautes fréquences qui ne sont pas utiles pour la représentation de la forme (ces coefficients représentent plutôt du bruit de quantification).

Compte tenu du nombre relativement faible d'exemples à notre disposition pour l'apprentissage (18 exemples par classe), nous conserverons seulement les coefficients d'ordre inférieur à 16 (ces coefficients décrivent encore relativement bien la forme). Ceci permet de limiter la dimension du réseau de neurones. En effet, un réseau sur-dimensionné risquerait de faire de l'apprentissage "par cœur", ce qui produirait une mauvaise généralisation. Mais il est bien entendu que si un plus grand nombre d'exemples étaient disponibles, il serait possible de conserver tous les coefficients d'ordre inférieur à 32. Remarquons toutefois que, à eux seuls, les coefficients d'ordre inférieur à 16 décrivent relativement bien la forme (fig 8).

4 Le réseau de neurones

4.1 Structure du réseau

Nous avons utilisé un perceptron multi-couches, entrainé par l'algorithme de rétropropagation, dans sa version améliorée (chap. 3). Le réseau est entièrement connecté entre 2 couches successives et est seuillé. La fonction de transition du neurone est la tangente hyperbolique (cf notations du chap. 2):

$$O_j = F(X_j) = th(X_j)$$

La couche d'entrée est alimentée avec les valeurs des descripteurs de Fourier normalisés d'ordre inférieur à 16.

Un réseau à coefficients complexes (chap. 3) a également été mis en œuvre. La fonction de transition est la suivante:

$$O_j = F(X_j) = th(|X_j|)e^{jArg(X_j)}$$

4.2 Apprentissage et Classification

Six classes ont été définies, et la couche de sortie du réseau contient 6 neurones (un par classe). Le tableau suivant indique le vecteur de sortie souhaité pour chacune des 6 classes:

Classe	Dénomination		Sorties souhaitées
1	SCOTCH	site=0^o	$(+1,-1,-1,-1,-1,-1)$
2	SCOTCH	site=30^o	$(-1,+1,-1,-1,-1,-1)$
3	STAPLER	site=0^o	$(-1,-1,+1,-1,-1,-1)$
4	STAPLER	site=30^o	$(-1,-1,-1,+1,-1,-1)$
5	WORKPIECE	site=0^o	$(-1,-1,-1,-1,+1,-1)$
6	WORKPIECE	site=30^o	$(-1,-1,-1,-1,-1,+1)$

L'apprentissage se fait conformément à l'algorithme décrit au chapitre 2 avec les améliorations du chapitre 3. Lorsque l'apprentissage est terminé, le réseau peut traiter de nouvelles données.

Pour la classification, le neurone dont la sortie est la plus forte détermine la classe.
Une mesure de confiance peut également être définie comme la moitié de l'écart entre la plus forte sortie et la sortie immédiatement inférieure (chap. 2).

5 Résultats expérimentaux

5.1 Représentation des entrées

Trois possibilités pour représenter les descripteurs de Fourier en entrée du réseau ont été envisagées:

Représentation 1 : Les descripteurs normalisés sont présentés sous forme cartésienne:

$$Re\{P_n\}$$

$$Im\{P_n\}$$

Représentation 2 : Seulement les modules des descripteurs normalisés sont présentées:

$$Module\{P_n\}$$

Représentation 3 : Les descripteurs normalisés sont présentés sous forme polaire:

$$100 * Module\{P_n\}$$

$$cos(argument\{P_n\})$$

$$sin(argument\{P_n\})$$

Le sinus et le cosinus de l'argument sont présentés à la place de l'argument lui-même, de façon à éviter le problème de discontinuité lors du passage de 359^o à 0^o. Le module est également multipliée par 100 afin de l'amener à l'ordre de grandeur des sinus et cosinus. Cette multiplication par 100 ne change pas les performances obtenues, mais permet d'accélérer l'apprentissage.

La silhouette est décrite par les coefficients P_{-16} à P_{-1} et P_2 à P_{16}, soit un total de 31 coefficients. La taille de la couche d'entrée du réseau de neurones sera donc 62, 31 ou 93 en fonction de la représentation choisie.

Pour information, les modules des descripteurs de Fourier pour les images de

l'ensemble d'évaluation sont représentés sur la figure 9. Les modules sont représentés en luminance (la brillance est proportionnelle à la racine carrée du module). En abscisse, on a l'indice du descripteur (2 à 16, puis -16 à -1), et en ordonnée on a le numéro de l'image. De gauche à droite et de haut en bas, on a les 18 exemples de la classe 0, puis les 18 exemples de la classe 1, etc. Cette figure permet de mieux apprécier la difficulté du problème. Aucune classe ne se distingue en effet clairement des autres, si ce n'est l'agrafeuse sous 0^o d'élévation (classe 3) par l'importance des hautes fréquences.

5.2 Performances obtenues

Les tableaux ci-dessous permettent de comparer les résultats obtenus avec différents réseaux de neurones: un réseau à 2 couches, un réseau à 3 couches (9 neurones sur la couche intermédiaire, automate seuil non compris) et un réseau à 4 couches (9 et 6 neurones sur les couches intermédiaires). A titre de comparaison, les résultats obtenus en remplaçant le réseau de neurones par un classifieur aux k-Plus-Proches-Voisins (k-ppv) sont fournis. Les classifieurs sont rangés par ordre décroissant des taux de généralisation:

Représentation cartésienne:

Méthode	Apprentissage	Généralisation	Nb de multiplications
4 couches	93%	70%	648
ppv	100%	65%	6696
3 couches	91%	63%	612
3-ppv	67%	59%	6696
2 couches	90%	57%	372
5-ppv	55%	52%	6696

Module:

Méthode	Apprentissage	Généralisation	Nb de multiplications
4 couches	94%	90%	369
3 couches	92%	86%	333
3-ppv	95%	84%	3348
2 couches	87%	83%	3348
ppv	100%	80%	186
5-ppv	90%	75%	3348

Représentation polaire:

Méthode	Apprentissage	Généralisation	Nb de multiplications
4 couches	96%	86%	927
3-ppv	94%	84%	10044
3 couches	95%	83%	891
ppv	100%	81%	10044
5-ppv	91%	77%	10044
2 couches	93%	72%	558

La matrice de confusion du réseau à 4 couches avec une représentation "module" est la suivante:

Classe	Effectif	Matrice de confusion					
0	18 (16.7%)	83%	17%	0%	0%	0%	0%
1	18 (16.7%)	0%	94%	0%	0%	6%	0%
2	18 (16.7%)	0%	0%	100%	0%	0%	0%
3	18 (16.7%)	0%	0%	6%	83%	0%	11%
4	18 (16.7%)	0%	5%	0%	6%	89%	0%
5	18 (16.7%)	0%	0%	0%	0%	11%	89%
Taux moyen de généralisation : **90%**							

En concaténant les lignes et colonnes correspondant à un même objet, on obtient la matrice de confusion suivante :

Classe	Effectif	Matrice de confusion		
0-1	36 (33.3%)	97%	0%	3%
2-3	36 (33.3%)	0%	94%	6%
4-5	36 (33.3%)	3%	3%	94%
Taux moyen de généralisation : **95%**				

Enfin, dans le cas à 6 classes, en rejetant les formes pour lesquelles la confiance est inférieure à 0.33, on obtient les résultats suivants sur l'ensemble d'évaluation:

correct : 81%
rejet : 17%
erreur : 2%

Il est donc possible de faire passer le taux d'erreur de 10% à 2%, au prix d'une non-décision dans 17% des cas.

6 Conclusion

Nous avons spécifié et expérimenté une méthode de reconnaissance d'objets 3D qui a l'avantage d'être simple à mettre en œuvre. En dépit d'une segmentation de mauvaise qualité, les résultats obtenus sont satisfaisants, ce qui prouve la robustesse de la méthode face à un prétraitement ou une acquisition déficients. Le système proposé est facile à implémenter et à utiliser car il n'y a ni modélisation, ni stéréoscopie, ni reconstruction 3D (une seule vue est suffisante). De plus la reconnaissance est très rapide.

Des expérimentations futures seront menées sur un plus grand nombre d'images. En particulier, la représentation polaire nous semble préférable à la représentation sous forme de module uniquement, car elle n'induit pas de perte d'information. Le fait que cette représentation n'ait pas conduit aux meilleurs résultats dans nos expérimentations nous semble plutôt dû au faible nombre d'exemples disponibles (ce qui favorise les réseaux de faible taille).

La représentation cartésienne semble être difficile à exploiter par le réseau. Ceci confirme l'idée intuitive selon laquelle il est plus facile d'interpréter un module et un argument qu'une partie réelle et une partie imaginaire. Les faibles performances du 3-ppv et du 5-ppv en apprentissage donnent d'ailleurs une indication quant à la

complexité de cette représentation. En effet, ces performances signifient que les plus proches voisins d'un exemple appartiennent souvent à des classes différentes de la sienne.

Annexe 1: Quelques propriétés des descripteurs de Fourier

Propriété 1 :

Le coefficient C_n est le nombre complexe qui minimise $L = \sum_{m=0}^{N-1} |U_m - C_n e^{j2\pi\frac{nm}{N}}|^2$

Démonstration :

$$\begin{aligned} |U_m - C_n e^{j2\pi\frac{nm}{N}}|^2 &= \left\{U_m^R - C_n^R cos(2\pi\frac{nm}{N}) + C_n^I sin(2\pi\frac{nm}{N})\right\}^2 \\ &\quad + \left\{U_m^I - C_n^R sin(2\pi\frac{nm}{N}) - C_n^I cos(2\pi\frac{nm}{N})\right\}^2 \end{aligned}$$

$$\begin{aligned} \frac{\partial L}{\partial C_n^R} &= -2\sum_{m=0}^{N-1}\left\{U_m^R - C_n^R cos(2\pi\frac{nm}{N}) + C_n^I sin(2\pi\frac{nm}{N})\right\} cos(2\pi\frac{nm}{N}) \\ &\quad -2\sum_{m=0}^{N-1}\left\{U_m^I - C_n^R sin(2\pi\frac{nm}{N}) - C_n^I cos(2\pi\frac{nm}{N})\right\} sin(2\pi\frac{nm}{N}) \\ &= -2\sum_{m=0}^{N-1}\left\{U_m^R cos(2\pi\frac{nm}{N}) + U_m^I sin(2\pi\frac{nm}{N}) - C_n^R\right\} \end{aligned}$$

$$\begin{aligned} \frac{\partial L}{\partial C_n^I} &= 2\sum_{m=0}^{N-1}\left\{U_m^R - C_n^R cos(2\pi\frac{nm}{N}) + C_n^I sin(2\pi\frac{nm}{N})\right\} sin(2\pi\frac{nm}{N}) \\ &\quad -2\sum_{m=0}^{N-1}\left\{U_m^I - C_n^R sin(2\pi\frac{nm}{N}) - C_n^I cos(2\pi\frac{nm}{N})\right\} cos(2\pi\frac{nm}{N}) \\ &= 2\sum_{m=0}^{N-1}\left\{U_m^R sin(2\pi\frac{nm}{N}) - U_m^I cos(2\pi\frac{nm}{N}) + C_n^I\right\} \end{aligned}$$

L'annulation des dérivées partielles conduit à :

$$C_n^R = \frac{1}{N}\sum_{m=0}^{N-1}\left\{U_m^R cos(2\pi\frac{nm}{N}) + U_m^I sin(2\pi\frac{nm}{N})\right\}$$

$$C_n^I = \frac{1}{N}\sum_{m=0}^{N-1}\left\{-U_m^R sin(2\pi\frac{nm}{N}) + U_m^I cos(2\pi\frac{nm}{N})\right\}$$

C'est à dire:

$$C_n = \frac{1}{N}\sum_{m=0}^{N-1} U_m e^{j2\pi\frac{nm}{N}}$$

Conséquence:
Le choix du sens trigonométrique comme sens de parcours du contour de l'objet implique que la norme du coefficient C_1 est le rayon du cercle qui approxime au mieux la forme (au sens des moindres carrés). Si l'on avait choisi le sens des aiguilles d'une montre, c'est le coefficient C_{-1} qui aurait joué ce rôle. Ceci justifie notre choix de normalisation via une division par C_1 et non par C_{-1}, car le coefficient C_1 sera généralement de norme importante, et la normalisation sera donc plus robuste.

Propriété 2:
En regroupant les coefficients de signes opposés, on peut voir la méthode des descripteurs de Fourier comme une décomposition de la forme par des ellipses.

Démonstration :
U_m est une somme de termes de la forme

$$U_{mn} = |C_n| e^{j(\phi_n + 2\pi \frac{nm}{N})} + |C_{-n}| e^{j(\phi_{-n} - 2\pi \frac{nm}{N})}$$

Lorsque m varie, on décrit une ellipse. Le grand axe est obtenu lorsque l'on est en phase, c'est à dire pour m tel que:

$$4\pi \frac{nm}{N} = \phi_n - \phi_{-n}$$

La norme de U_{mn} vaut alors $|C_n| + |C_{-n}|$. Le petit axe est obtenu lorsque l'on est en opposition de phase, c'est à dire pour m tel que:

$$4\pi \frac{nm}{N} = \phi_n - \phi_{-n} + \pi$$

La norme de U_{mn} vaut alors $||C_n| - |C_{-n}||$.

Références

[*DeGreef*89] Bart DE GREEF
"Classical approaches to 3 dimensional object recognition"
Esprit II project n^o2059 "Pygmalion", report R36-2, September 1989

[*DeGreef*90] Bart DE GREEF
"Prototype implementation of a classical approach to 3D object recognition"
Esprit II project n^o2059 "Pygmalion", report R36-4, June 1990

[*Guerra*90] C. GUERRA, E.N. HOUSTIS
"Definition of algorithms for 3D pattern recognition"
Esprit II project n^o2059 "Pygmalion", report R35-1, January 1990

[*Granlund*72] G.H. GRANLUND
"Fourier processing for hand print character recognition"
IEEE Trans. Computers, vol 21, pp 195-201, 1972

[*Lin*87] C.S. LIN, C.L. HWANG
"New forms of shape invariants from elliptic Fourier Descriptors"
Pattern Recognition, vol 20, n^o5, pp 535-545, 1987

[*Mitchell*83] O.R. MITCHELL, T.A. GROGEN
"Evaluation of Fourier Descriptors for target recognition in digital imagery"
Purdue University, report RADC-TR-83-33, February 1983

[*Rumelhart*86] D.E. RUMELHART, G.E. HINTON, R.J. WILLIAMS
"Learning internal representations by error backpropagation"
Parallel Distributed Processing, D.E. RUMELHART and J.L. Mc CLELLAND
Chap 8, Bradford Book - MIT Press - 1986

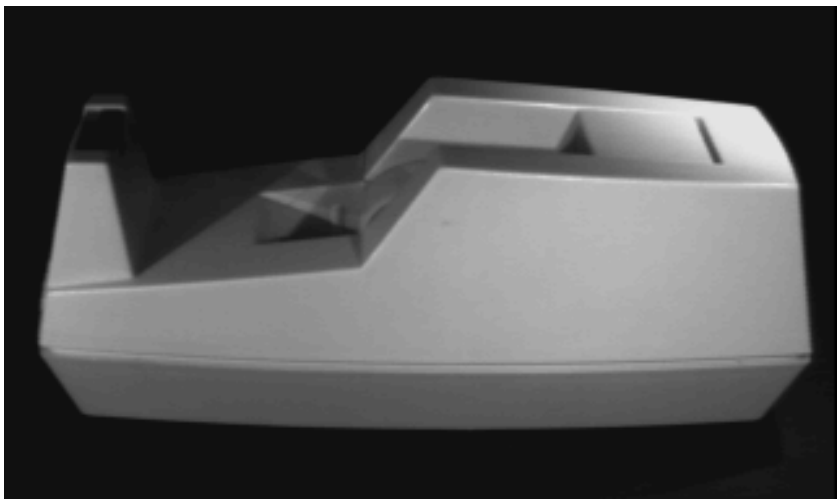


Figure 1: SCOTCH, site=30^o, azimut=270^o

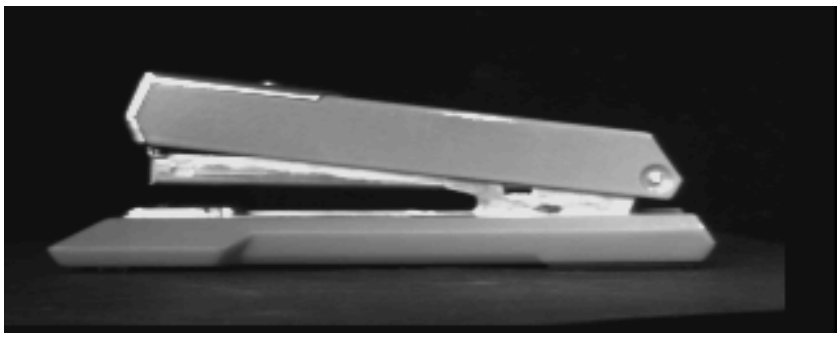


Figure 2: STAPLER, site=0^o, azimut=90^o

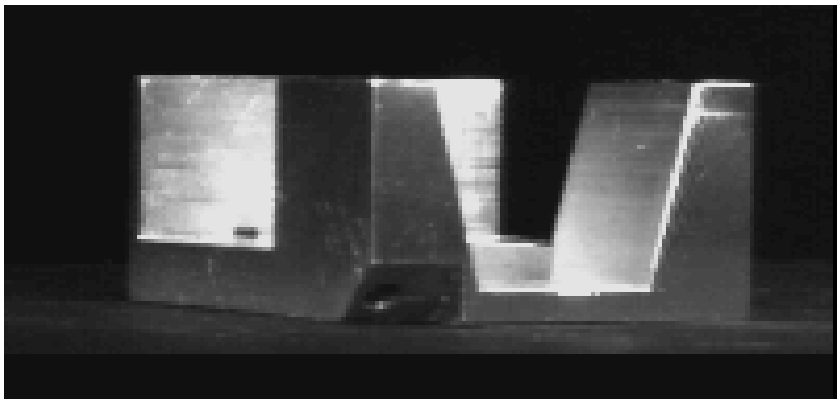


Figure 3: WORKPIECE, site=0^o, azimut=150^o

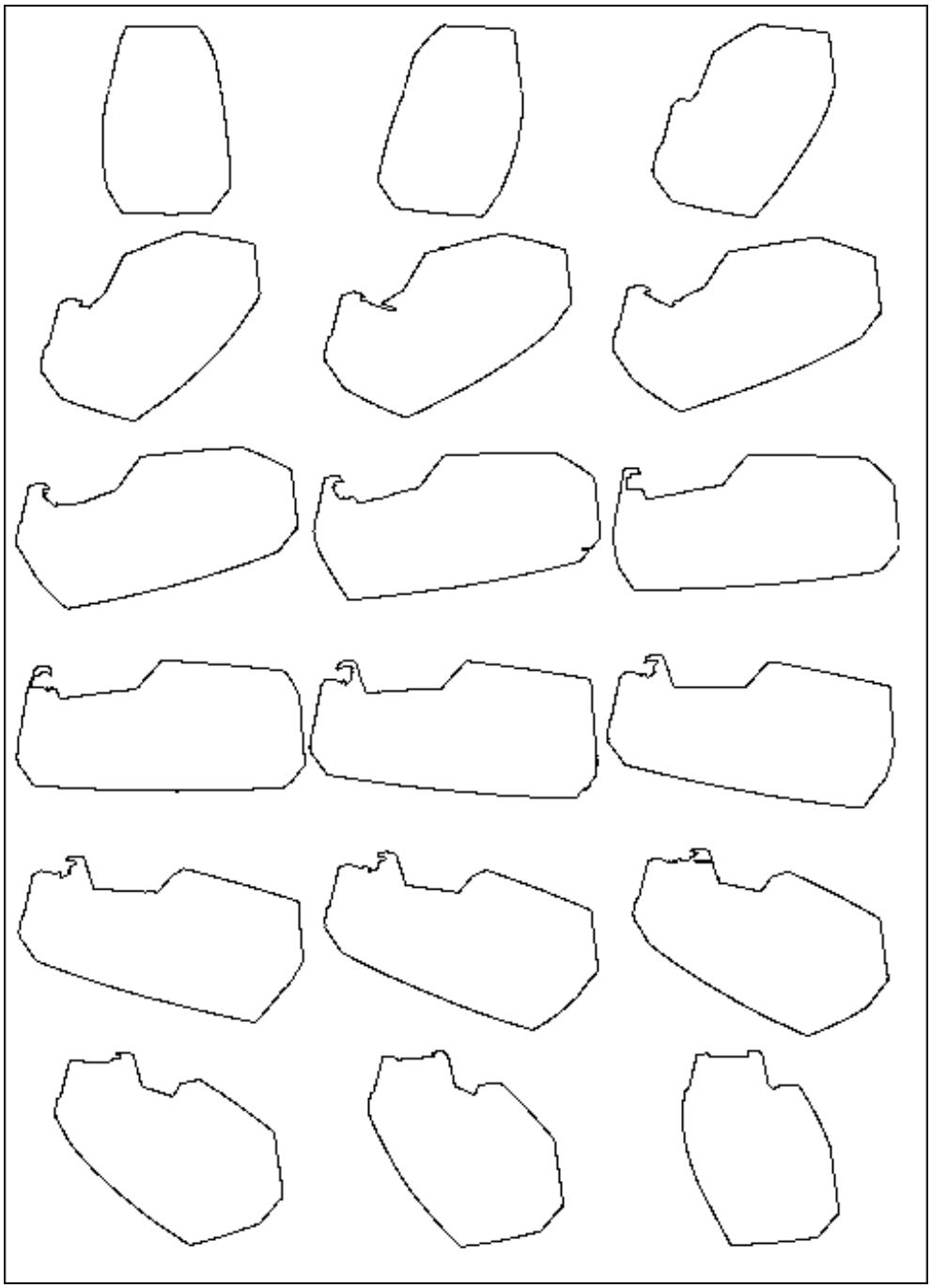


Figure 4: Contours de SCOTCH, site=30^o, azimut=180^o à 350^o

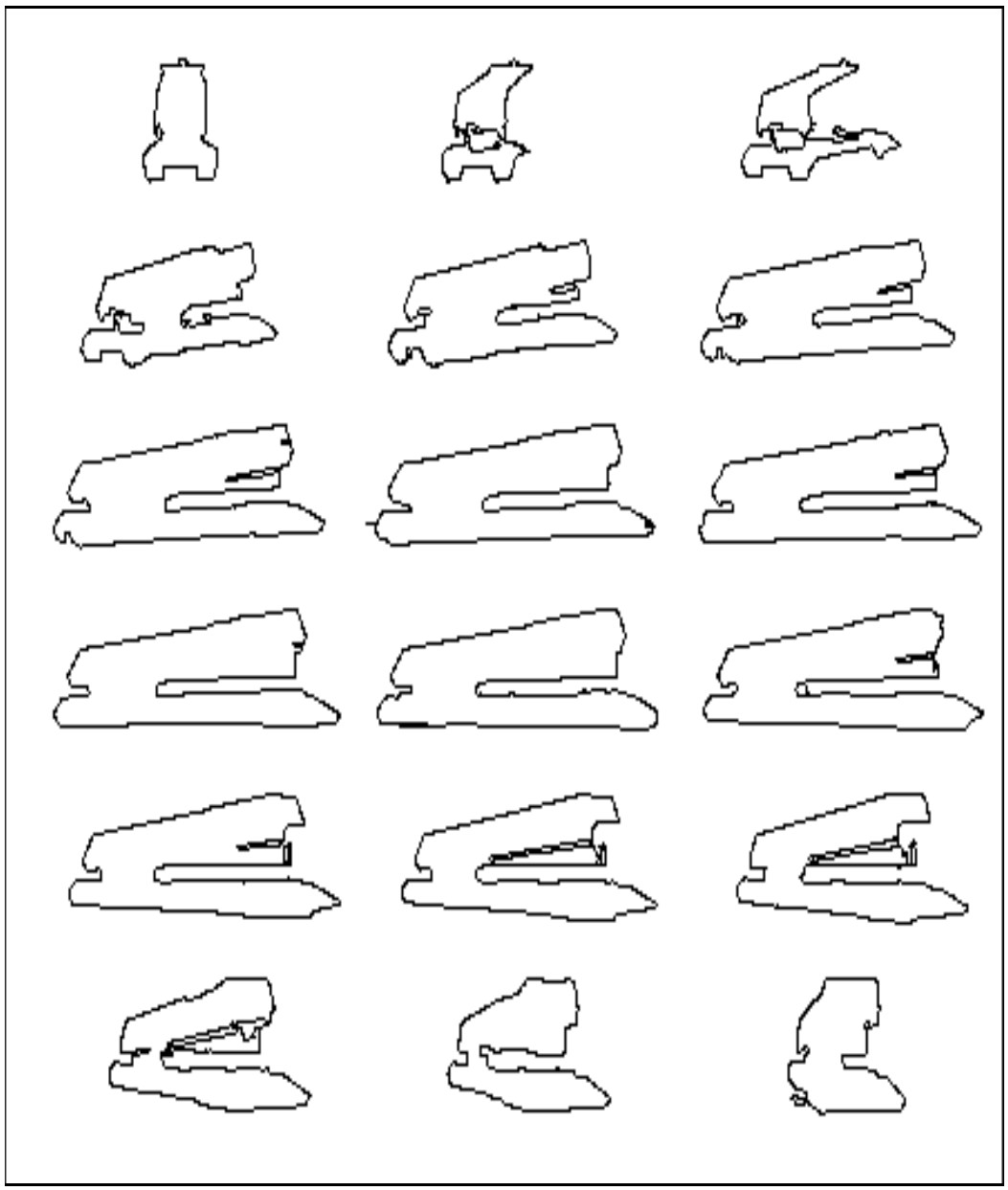


Figure 5: Contours de STAPLER, site=0^o, azimut=180^o à 350^o

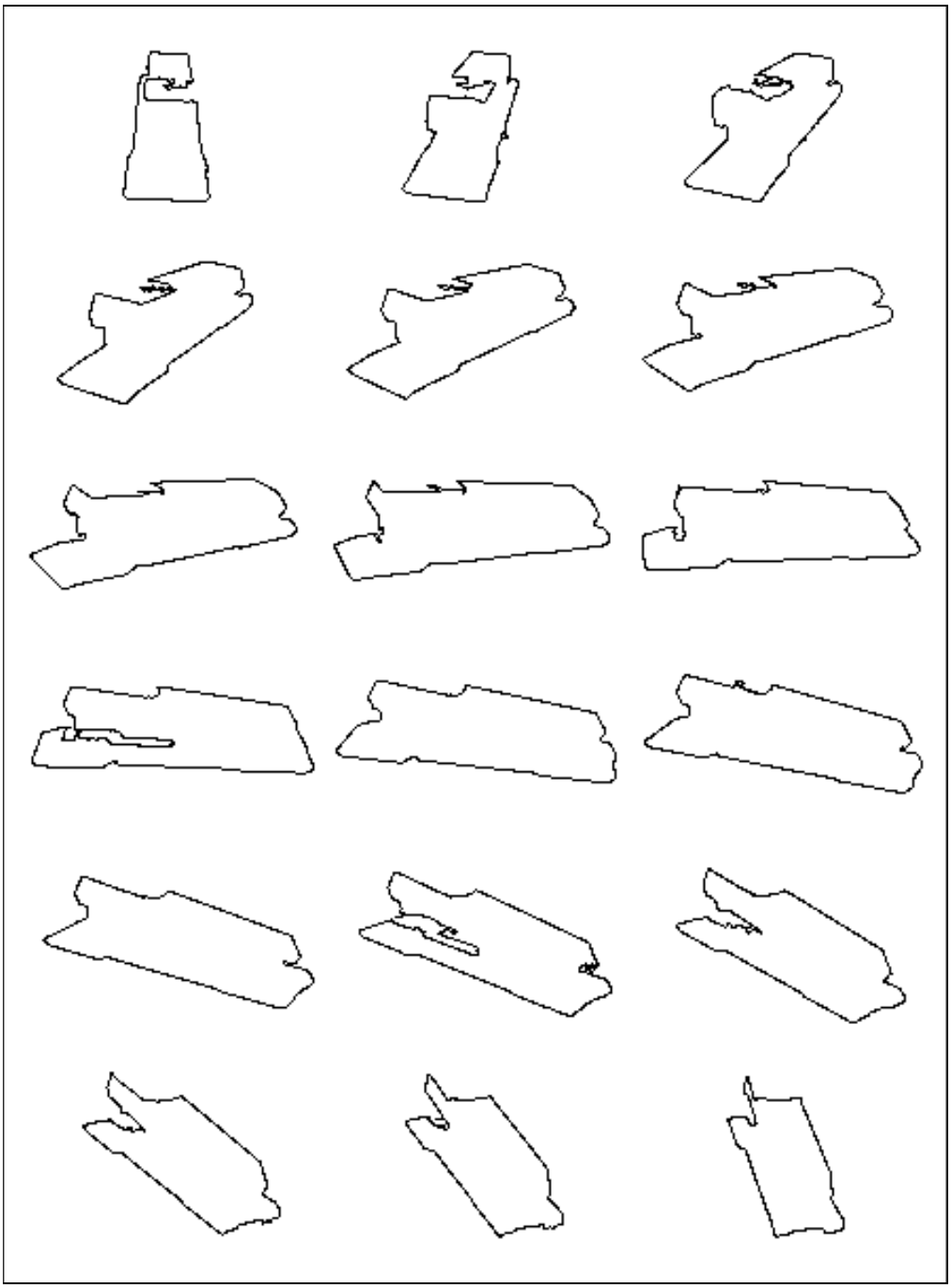


Figure 6: Contours de STAPLER, site=30^o, azimut=0^o à 170^o

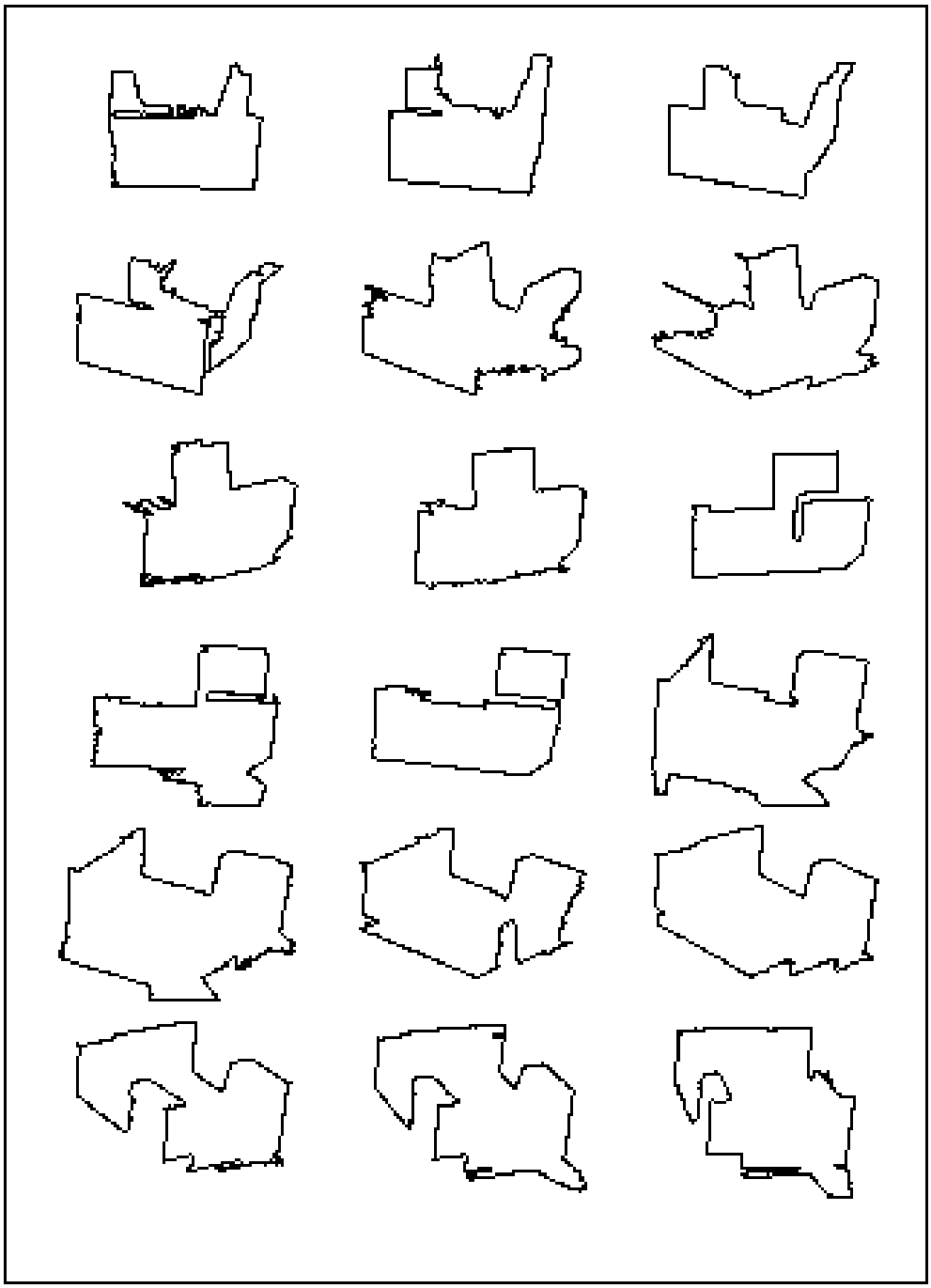


Figure 7: Contours de WORKPIECE, site=30^o, azimut=0^o à 170^o

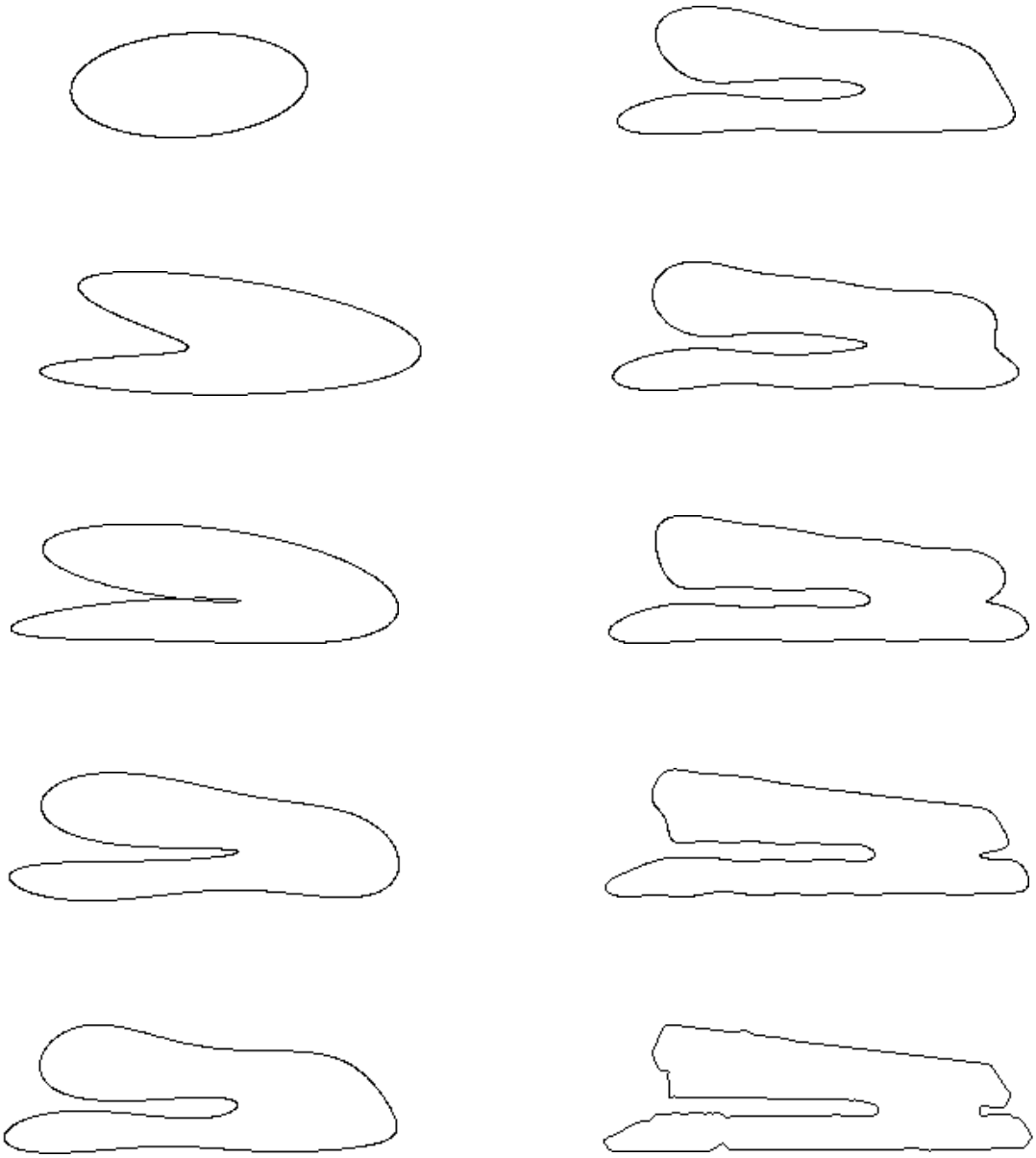


Figure 8: Reconstruction de l'agrafeuse ($\Delta = 1, 2, 4, 5, 6, 8, 10, 16, 32$) et original

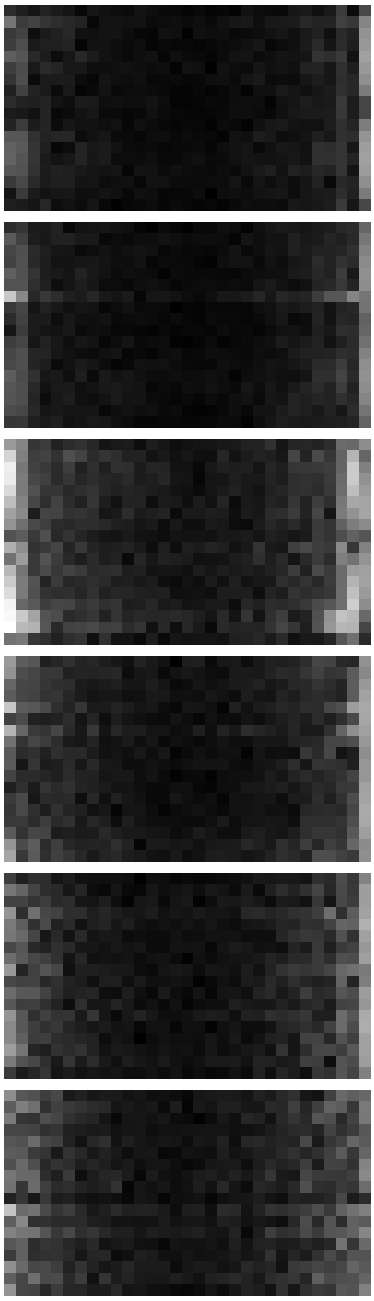


Figure 9: Modules des descripteurs de Fourier (ensemble d'évaluation, racine carrée). *En abscisse: l'ordre du coefficient (2 à 16 puis -16 à -1). En ordonnée: le numéro de l'exemple (de haut en bas: classes 1 à 6).*

Chapitre 8:

RECONNAISSANCE DE CHIFFRES MANUSCRITS

La reconnaissance de l'écriture a été l'un des premiers domaines d'application des réseaux de neurones. Nous nous intéressons ici plus particulièrement à la reconnaissance de chiffres manuscrits. Des méthodes neuronales performantes ont déjà été proposées pour réaliser cette tâche.

Nous proposons une approche intermédiaire entre les méthodes classiques, qui sont généralement basées sur l'extraction d'un petit nombre de caractéristiques discriminantes, et les approches "tout-neuronal", dans lesquelles un réseau de neurones est directement alimenté par l'image du chiffre. Des résultats expérimentaux sur une base de données de 2589 chiffres, provenant de 30 scripteurs différents, sont présentés.

1 Introduction

La reconnaissance de l'écriture manuscrite a fait l'objet de nombreuses recherches au cours des 30 dernières années [*Govindan*90]. Les applications potentielles sont par exemple:

- La reconnaissance de codes postaux [*Mitchell*89]
- Le traitement de bons de commande
- La robotique (tri de pièces marquées manuellement) [*Beauville*90]
- Le traitement de chèques bancaires

Pour une revue, on pourra consulter par exemple [*Govindan*90], [*Ross*88], [*Tappert*90]. Les diverses recherches peuvent être classifiées en fonction des critères suivants:

- Reconnaissance "On-Line"(en ligne) ou "Off-Line" (optique). Dans le premier cas, on connait les mouvements du crayon (l'écriture se fait par exemple sur une tablette graphique), alors que dans le second cas on a uniquement l'image du caractère ou du texte. Les taux de reconnaissance sont donc meilleurs en "On-Line".
- Reconnaissance avec scripteur connu ou inconnu: dans le premier cas, la personne qui écrit est connue (ou appartient à un groupe connu, d'une dizaine de personnes par exemple). On peut donc adapter le système de reconnaissance à ce groupe.

Nous nous intéresserons ici à la reconnaissance de chiffres manuscrits en "Off-Line", et sans contrainte d'écriture (une application typique est la reconnaissance de codes postaux). Le problème de la séparation des chiffres connectés ne sera pas étudié. Cette séparation peut être réalisée en utilisant par exemple la méthode proposée dans [*Shridhar*87]. Dans le cadre de notre étude, nous traiterons donc des chiffres préalablement isolés.

2 Travaux antérieurs

La reconnaissance de chiffres manuscrits en utilisant des méthodes classiques a été traitée par divers auteurs:

- Mitchell [*Mitchell*89] utilise une méthode de mise en correspondance avec des modèles, ces modèles décrivant la disposition spatiale de diverses primitives morphologiques (trous, cavités, ...). Les modèles sont créés interactivement (il n'y a pas d'apprentissage). Les performances annoncées, sur une base de données de 2103 chiffres sont de 88.9% de reconnaissance, 10.1% de rejet, et 1.0% d'erreur.
- Shridhar [*Shridar*86] utilise un système à base de règles de décision. Ces règles sont formées à partir de tests sur les valeurs d'un petit nombre de caractéristiques extraites des profils gauche et droit des chiffres (positions des extréma, ...). Ces règles ont été construites par l'auteur afin d'obtenir de bons résultats sur sa base de données. Sur une base de données de 500 chiffres, un taux de reconnaissance de 99% est annoncé.

La comparaison des performances annoncées par différents auteurs est difficile car les bases de données sont différentes. De plus la majorité des auteurs élaborent leur système sur la même base de données que celle qui sert à l'évaluation (dans le cadre neuronal, ceci reviendrait à faire l'évaluation sur l'ensemble d'apprentissage). Les performances annoncées peuvent donc approcher à volonté des 100%, mais il est très difficile de prévoir ce que seraient ces performances sur de nouvelles données.

Dans le domaine neuronal, un certain nombre de travaux ont été publiés au cours des dernières années. Le réseau de neurones est en général directement alimenté par l'imagette du chiffre binarisé, normalisée à une certaine taille (de l'ordre de 16x16 pixels).

- Stringa [*Stringa*89] présente directement l'imagette 16x32 du chiffre en entrée du réseau. Ce réseau est d'un type particulier, car il s'agit d'un ensemble de fonctions booléennes, qui sont élaborées pendant l'apprentissage sur des critères statistiques. Sur une base de 20000 chiffres (10000 pour l'apprentissage et 10000 pour l'évaluation), les performances en généralisation sont de 79% de reconnaissance, 15% de rejet, et 6% d'erreur.
- Pawlicki [*Pawlicki*88] a comparé différents modèles de réseaux de neurones (mémoires associatives, perceptron multicouches, Hopfield, machine de Boltzmann,

...) lorsqu'ils sont alimentés par l'imagette 16x16 du chiffre. L'apprentissage a été réalisé sur 391 chiffres, et l'évaluation a eu lieu sur 1173 chiffres. Les meilleures performances (74% de reconnaissance) ont été obtenues avec une mémoire associative linéaire, c'est à dire un réseau très simple. Mais il est probable que ce résultat soit dû à un phénomène de sur-apprentissage (apprentissage "par coeur") des réseaux de type Boltzmann ou perceptron multicouches car l'ensemble d'apprentissage est de faible taille.

- Le Cun [*LeCun*89] utilise un perceptron 5 couches à connexions locales entrainé par l'algorithme de rétropropagation, et alimenté par l'imagette 16x16 du chiffre. Il contient 1256 neurones et 64660 connexions (dont seulement 9760 sont indépendantes car des contraintes d'égalité sont imposées entre certaines connexions). Sur une base de données de 9298 chiffres (7291 pour l'apprentissage, et 2007 pour l'évaluation), le taux d'erreur en généralisation est de 5%. Le temps d'apprentissage est d'environ 3 jours sur Sun4. Cette méthode est actuellement considérée comme étant la meilleure, méthodes neuronales et classiques confondues.

3 Méthode proposée

La méthode que nous proposons repose sur les idées suivantes:

1. Il n'est pas forcément intéressant d'alimenter directement le réseau par l'imagette du chiffre, car on le force alors à réaliser une tâche très complexe, ce qui suppose un très grand nombre d'exemples, un réseau de taille importante, et un apprentissage très long.
2. D'un autre coté, il peut être dangereux d'alimenter le réseau avec seulement un petit nombre de caractéristiques, du type de celles qui sont extraites dans les méthodes classiques, car, si l'apprentissage s'en trouve facilité, on risque d'être confronté à un problème de recouvrement de classes lorsque la base de données devient importante.
3. L'utilisation de connexions locales et de contraintes sur les poids (lorsque le type de données s'y prête) favorise une bonne généralisation car on diminue le nombre de paramètres libres dans le réseau.

L'approche que nous proposons consiste à alimenter un réseau multicouches par les courbes représentant les profils des chiffres, ainsi que par quelques autres

caractéristiques susceptibles de lever certaines ambiguïtés résiduelles. On ne perd donc pas (ou très peu) d'information (cf point n^o2), et on facilite la tâche du réseau par rapport à une alimentation directe par l'image (cf point n^o1). Enfin, conformément au point n^o3, nous mettrons en œuvre des connexions locales et nous proposerons un algorithme d'apprentissage qui permet d'introduire une notion de voisinage entre certains neurones, et donc des contraintes sur les poids.

4 Les prétraitements

4.1 Les caractéristiques métriques

Les caractéristiques métriques mises en œuvre sont les profils des chiffres. L'utilisation des profils pour caractériser les chiffres manuscrits a été suggérée par divers auteurs ([*Shridhar*86],[*Gokana*86] par exemple). Le chiffre est encadré dans une fenêtre minimale. Les profils gauche et droit du chiffre sont extraits sur l'image seuillée (fig 1). Ensuite, des paramètres sont calculés sur les profils: Shridhar calcule des paramètres tels que la position du maximum, Gokana réalise une description symbolique de l'approximation polygonale des profils.

Ces méthodes consistent donc en l'extraction d'un petit nombre de paramètres bien choisis, supposées suffisants pour discriminer les caractères. Shridhar annonce un taux de reconnaissance excellent (99%), ce qui permettrait de penser que le choix a été dans ce cas particulièrement pertinent. Toutefois il convient d'être prudent quant à l'interprétation de ces résultats car le choix des paramètres et des règles de décision a été élaboré sur la même base de données que celle qui a servi à l'évaluation.

L'approche que nous proposons est différente. Elle consiste à alimenter directement un réseau de neurones avec les profils. On s'affranchit ainsi du problème de choix de paramètres discriminants, et on limite les risques de recouvrement de classes lorsque la base de données devient importante, car on reste dans un espace de grande dimension et on conserve toute l'information.

Toutefois, dans certains cas les profils gauche et droit peuvent être insuffisants pour discriminer les chiffres (fig 2). Afin de lever certaines ambiguités, on alimentera également le réseau de neurones avec un profil orienté à 45^o (fig 1). Au niveau logiciel, on peut utiliser pour le calcul de ce profil le même module que pour le calcul du profil gauche, si l'on fait simplement subir au préalable au chiffre une rotation de -45^o.

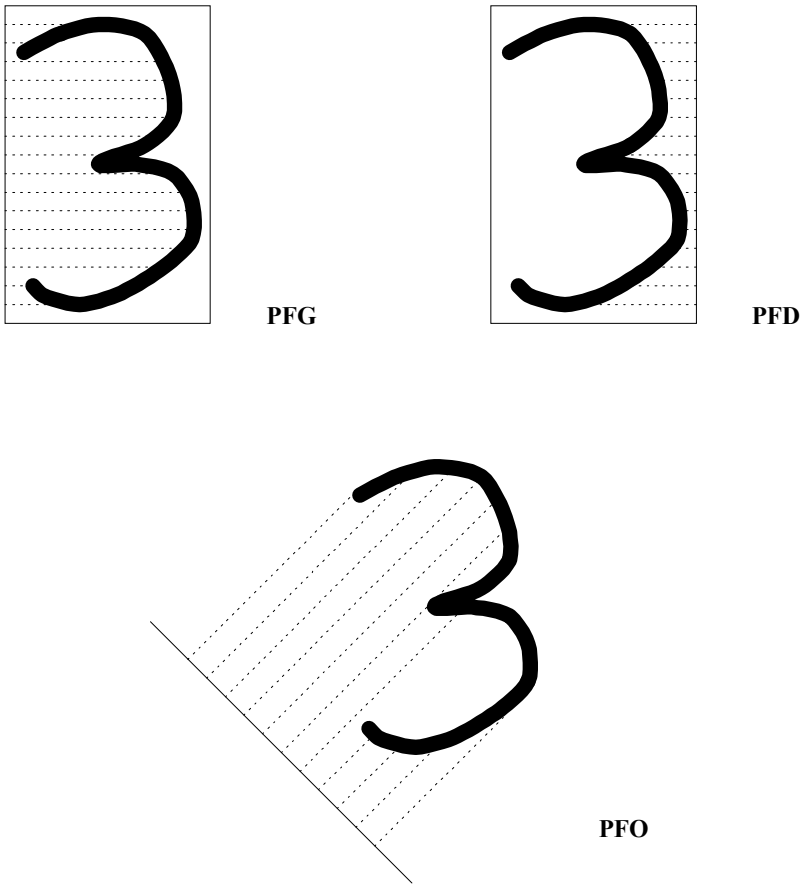


Figure 1: *Les profils gauche, droit, et orienté*

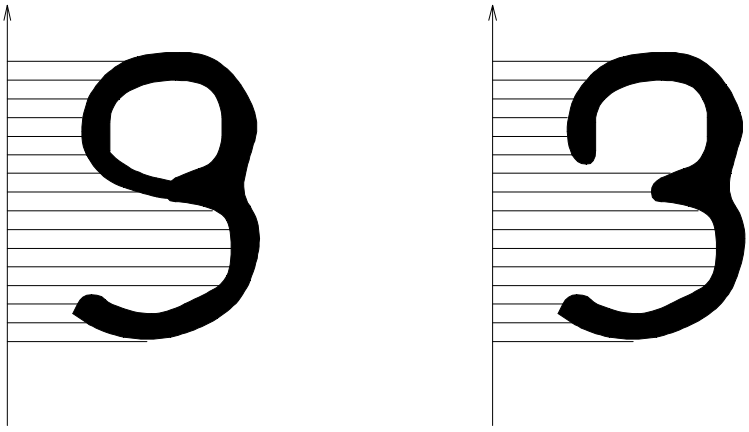


Figure 2: *Exemple de confusion sur les profils gauche et droit*

On réalise une interpolation linéaire du profil dans les zones où le caractère est cassé (par exemple un "5" dont la barre supérieure est déconnectée du reste du

chiffre). Enfin, les différents profils sont normalisés afin d'obtenir l'invariance par rapport aux dilatations du chiffre. On ramène le nombre d'échantillons à 32 pour les profils gauche et droit, et à 16 pour le profil orienté, ceci étant par exemple réalisé par interpolation linéaire. Le nombre d'échantillons pour le profil orienté est plus faible (16 au lieu de 32) car il s'agit d'une information complémentaire. Puis on ramène la valeur de chaque échantillon entre -1 et +1 (fig 3).

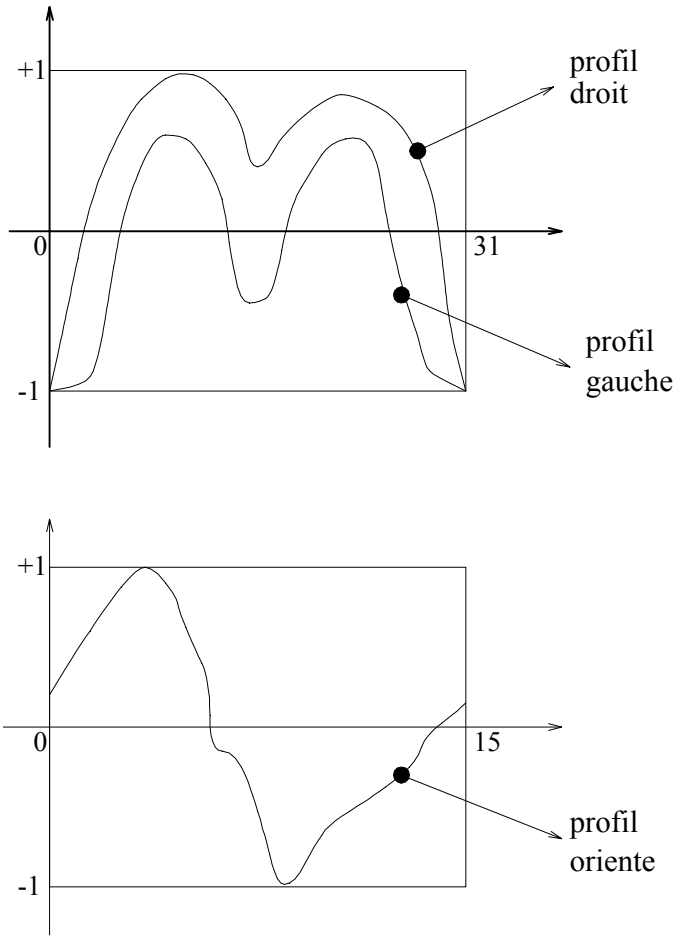


Figure 3: *normalisation des profils*

4.2 Les caractéristiques statistiques

Une information globale caractérisant la répartition des points constituant le chiffre dans la fenêtre minimale est également exploitée. La fenêtre englobant le chiffre étudié est divisée en régions de 6 façons différentes (fig 4). Pour chaque découpage, on détermine la répartition des points du chiffre entre les régions constituant ce découpage.

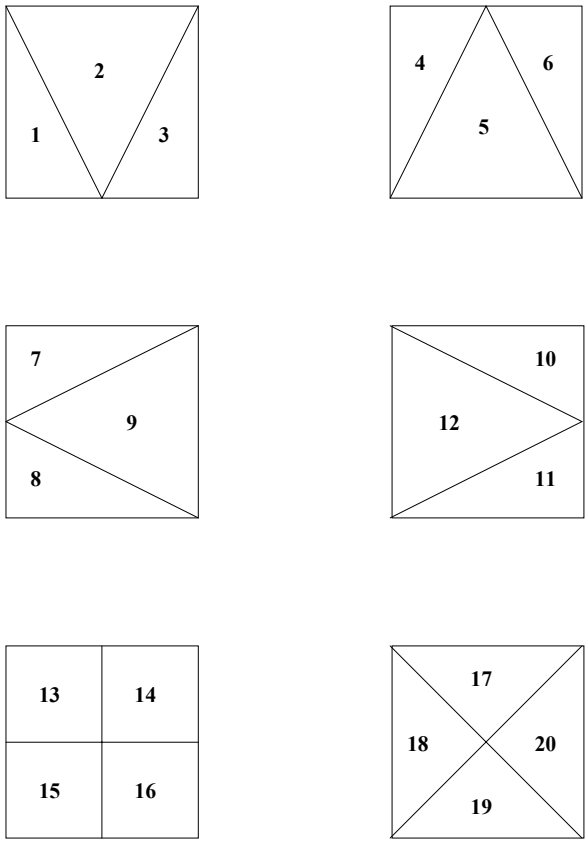


Figure 4: *Les 20 zones définies par 6 masques*

Ce type de caractéristiques a été proposé dans [*Beauville*90] pour un problème mono-scripteur. Ces caractéristiques semblent en effet bien discriminantes pour un scripteur donné. Leur validité pour notre problème où le scripteur est quelconque et inconnu n'est pas évidente a priori. Toutefois, comme nous le verrons dans la section consacrée aux résultats expérimentaux, leur prise en compte a permis d'améliorer légèrement les performances de notre système.

Au total, on a 20 régions, d'où 20 caractéristiques R_j telles que $R_j = n_j/n$, où n_j est le nombre de pixels du chiffre dans la région j, et n le nombre total de pixels du chiffre.

4.3 Les caractéristiques morphologiques

Un dernier type de caractéristique pris en compte est la surface relative de diverses cavités et trous. On définit 5 types de cavités: les cavités Ouest, Est, Nord, Sud,

et Centrales. Avec les trous, on a 6 primitives morphologiques (fig 5). Ce type de primitives morphologiques a par exemple été utilisé par Mitchell [*Mitchell*89] pour construire un modèle symbolique du chiffre (le modèle contient la disposition spatiale relative des différentes cavités et trous).

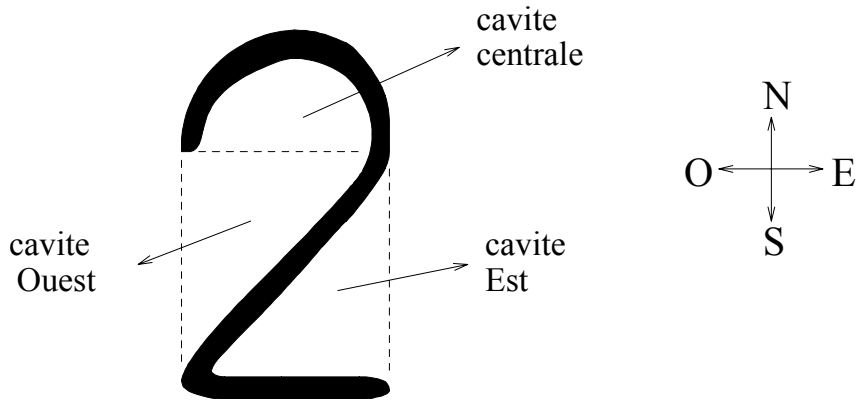


Figure 5: *Primitives morphologiques*

Les cavités Ouest sont définies de la façon suivante:

Un point appartient à une cavité Ouest si:

- En se déplacant en ligne droite dans les directions Est, Sud ou Nord, on rencontre le chiffre.
- En se déplacant en ligne droite dans la direction Ouest, on ne rencontre pas le chiffre.
- Ce point n'appartient pas au chiffre.

Une définition similaire est mise en œuvre pour les cavités Nord, Sud et Est. Les cavités centrales sont définies de la façon suivante:

Un point appartient à une cavité Centrale si:

- En se déplacant en ligne droite dans les directions Ouest, Est, Sud ou Nord on rencontre le chiffre.
- Ce point n'appartient pas au chiffre.
- Ce point n'appartient pas à un trou.

Enfin, Les trous peuvent être détectés à l'aide d'un algorithme de recherche de zones connexes. En effet, si l'on recherche les zones connexes relatives au fond, le chiffre étant placé dans une fenêtre avec laquelle il n'est pas en contact, on trouvera :

- Une grande zone connexe en contact avec les bords de la fenêtre. Elle correspond au fond externe au chiffre.
- Eventuellement, une ou plusieurs zones non en contact avec les bords de la fenêtre : elles correspondent aux trous.

Chacune de ces 6 primitives pourrait être représentée en entrée du réseau de neurones par la surface relative de la primitive par rapport à la surface totale de toutes les primitives. Afin de tenir compte de certains cas particuliers (par exemple le "1" représenté par un trait vertical), on force à 0 ces valeurs lorsque la surface totale de toutes les primitives est inférieure à la moitié de la surface du chiffre.

Une amélioration consiste à représenter chaque primitive par 2 valeurs. Il s'agit toujours de surfaces relatives, mais pondérées par les fonctions G_h et G_b de la figure 6. Ainsi, par exemple, les cavités Ouest seront représentées par les 2 valeurs suivantes:

$$CO_h = \frac{\sum_{lignes}\{\sum_{colonnes} I_{CO}(ligne, colonne)G_h(ligne)\}}{\sum_{lignes}\{\sum_{colonnes} I_{CT}(ligne, colonne)G_h(ligne)\}}$$

$$CO_b = \frac{\sum_{lignes}\{\sum_{colonnes} I_{CO}(ligne, colonne)G_b(ligne)\}}{\sum_{lignes}\{\sum_{colonnes} I_{CT}(ligne, colonne)G_b(ligne)\}}$$

Où I_{CO} est l'indicatrice des cavités Ouest, I_{CT} est l'indicatrice de toutes les cavités ou trous. Ceci permet de disposer d'une information relative à la disposition spatiale des primitives morphologiques.

5 Le réseau de neurones

5.1 Structure

Le réseau de neurones reçoit 112 entrées:

- 32 valeurs pour le profil gauche.

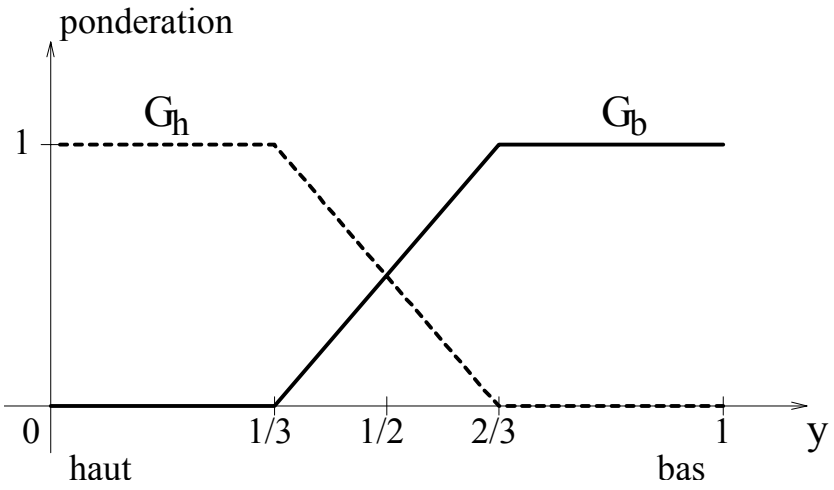


Figure 6: *Pondération des primitives morphologiques*

- 32 valeurs pour le profil droit.
- 16 valeurs pour le profil orienté.
- 20 valeurs pour les caractéristiques statistiques (zones).
- 12 valeurs pour les caractéristiques morphologiques.

Il serait possible de mettre en œuvre une structure entièrement connectée entre couches successives. Toutefois, il existe une notion de voisinage au niveau des points des profils, qu'il est intéressant de traduire dans la structure du réseau. Ceci peut être réalisé en utilisant des connexions locales, comme indiqué figure 7. Ainsi,chaque neurone de la première couche devient responsable d'une région particulière du ou des profils.

Le réseau proposé comporte 5 couches (fig 7). Les neurones seuils (neurones dont la sortie est toujours à 1) n'ont pas été représentés sur la figure. Ils sont présents sur les couches 1, 3, et 4.

- La première couche contient 112 unités d'entrée.
- La seconde couche contient 88 neurones:
  - 3 lignes de 6 neurones responsables du profil orienté. Chaque neurone reçoit 6 entrées correspondant à des points successifs du profil. Les entrées se

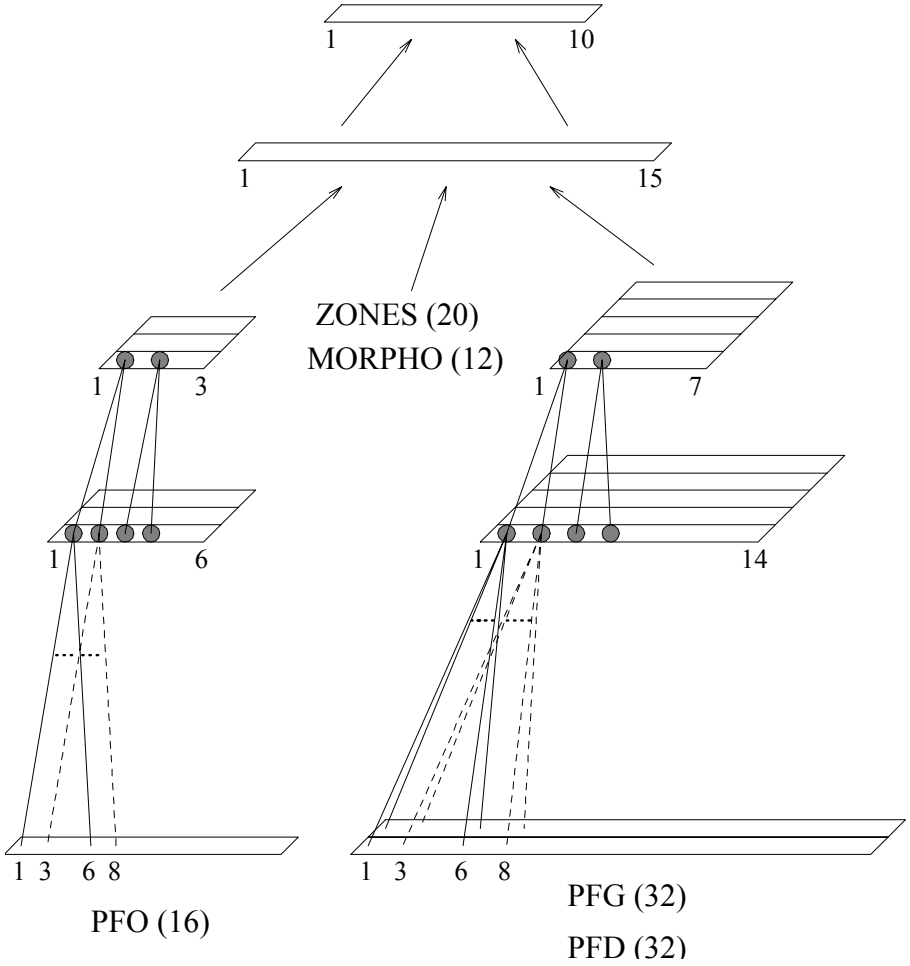


Figure 7: *La structure du réseau multicouches*

décalent par pas de 2 lorsque l'on passe d'un neurone à son voisin dans une ligne. Ainsi, le premier neurone de chaque ligne reçoit les entrées 1 à 6, le second les entrées 3 à 8, etc. Chaque neurone reçoit de plus une entrée provenant du neurone seuil.

- 5 lignes de 14 neurones responsables des profils gauche et droit. Chaque neurone reçoit 12 entrées (6 points successifs du profil gauche, et les 6 points correspondant sur le profil droit). Les entrées se décalent par pas de 2 lorsque l'on passe d'un neurone à son voisin dans une ligne. Chaque neurone reçoit de plus une entrée provenant du neurone seuil.

• La troisième couche contient 44 neurones:

- 3 lignes de 3 neurones responsables du profil orienté. Chaque neurone reçoit

2 entrées (2 neurones de la ligne qui lui correspond sur la couche précédente, comme indiqué sur la figure). On impose aux 2 poids d'être identiques. Ceci force en quelque sorte ces neurones à réaliser un moyennage de leurs entrées. On impose ainsi une invariance par rapport à de légères déformations locales des profils.

- 5 lignes de 7 neurones responsables des profils gauche et droit. Chaque neurone reçoit 2 entrées (2 neurones de la ligne qui lui correspond sur la couche précédente, comme indiqué sur la figure). On impose aux 2 poids d'être identiques.

- La quatrième couche contient 15 neurones. Elle reçoit 76 entrées (les 44 sorties des neurones de la couche 3, les 20 valeurs des caractéristiques statistiques, et les 12 valeurs des caractéristiques morphologiques). Chaque neurone reçoit de plus une entrée provenant du neurone seuil. Les connexions sont totales.

- La cinquième couche contient 10 neurones, correspondant aux 10 classes. Elle reçoit 15 entrées (les 15 sorties de la couche 4) et un seuil. Les connexions sont totales.

On peut à présent faire le total du nombre de liaisons:

- Entre les couches 1 et 2:
  - Pour le profil orienté :
    (3 lignes)x(6 neurones par ligne)x(7 entrées par neurone) = 126 liaisons
  - Pour les profils gauche et droit :
    (5 lignes)x(14 neurones par ligne)x(13 entrées par neurone) = 910 liaisons
- Entre les couches 2 et 3 :
  - Pour le profil orienté :
    (3 lignes)x(3 neurones par ligne)x(2 entrées par neurone) = 18 liaisons
  - Pour les profils gauche et droit :
    (5 lignes)x(7 neurones par ligne)x(2 entrées par neurone) = 70 liaisons
- Entre les couches 1, 3 et la couche 4 :

  (15 neurones)x(32+44+1 entrées par neurone) = 1155 liaisons.
- Entre les couches 4 et 5 :

  (10 neurones)x(15+1 entrées par neurone) = 160 liaisons.

Le nombre total de liaisons est donc de 2439.

5.2 L'algorithme d'apprentissage

L'algorithme de base est l'algorithme de rétropropagation, auquel nous avons apporté quelques modifications afin de permettre la définition d'un voisinage entre certains neurones d'une même couche.

Notons $W_{i,j}$ le poids de la liaison entre le neurone i et le neurone j, et $\Delta W_{i,j}(t)$ la variation du poids proposée par l'algorithme de rétropropagation à l'itération t. Alors, la variation du poids que nous réaliserons est:

$$\Delta_{\#} W_{i,j}(t) = \frac{1}{1 + \sum_{k \in V(j)} P(k,j)} \left\{ \Delta W_{i,j}(t) + \sum_{k \in V(j)} P(k,j) \Delta W_{i'(j,k,i),k}(t) \right\}$$

V(j) est le voisinage de j, et P(k,j) est un coefficient de pondération. i'(j,k,i) est le neurone qui joue par rapport à k le même rôle que i par rapport à j. Ceci suppose donc que les neurones j et k ont le même nombre de connexions en entrée.

Un voisinage est ainsi défini à l'intérieur de chaque ligne de neurones pour la couche 2 de la figure 7. Un neurone a pour voisins les 2 neurones les plus proches sur la même ligne (ou le neurone le plus proche pour les neurones situés aux extrémités). Le coefficient de pondération a été choisi égal à 0.9. Il est bien entendu possible de définir un voisinage plus étendu à l'intérieur de la ligne, avec par exemple une pondération gaussienne en fonction de la distance.

Enfin, si j et k sont voisins, les poids $W_{i,j}$ et $W_{i'(j,k,i),k}$ sont choisis égaux lors de l'initialisation.

Ce type d'apprentissage a pour effet de contraindre les poids de neurones voisins à rester voisins, d'où un effet comparable à une réduction du nombre de paramètres libres dans le réseau, et donc un apprentissage plus difficile, mais une meilleure généralisation. Toutefois cette contrainte n'est pas une stricte contrainte d'égalité: on conserve ainsi une possibilité de différenciation des neurones à l'intérieur d'une ligne d'où une possibilité d'adaptation plus grande.

L'idée intuitive sous jacente à ce type d'apprentissage est la suivante: on peut considérer un neurone, comme un filtre linéaire suivi d'un processus de décision (la fonction d'activation). Il est donc logique que si un filtre est bien adapté pour une zone du profil, le filtre bien adapté à une zone voisine en soit proche.

6 Résultats expérimentaux

6.1 Conditions expérimentales

Afin de disposer d'écritures assez diversifiées, 30 personnes des L.E.R. ont rempli chacune une page de chiffres (format A4). Les chiffres sont disposés par ligne afin de simplifier la recherche automatique des chiffres dans l'image (cette recherche automatique n'étant pas l'objet de notre étude), et de permettre un calcul automatique de la classe de chaque chiffre (le fichier d'apprentissage peut ainsi être réalisé sans l'intervention d'un opérateur).

Deux types d'expériences ont été menées:

1. **Problème multi-scripteurs inconnus** : Parmi les 30 pages, 16 ont été utilisées pour l'apprentissage et 14 pour l'évaluation (ce découpage ayant été réalisé aléatoirement). Ceci représente 1414 chiffres pour l'apprentissage et 1175 pour l'évaluation (les différentes classes sont approximativement équiprobables).
2. **Problème multi-scripteurs connus** : Pour chaque page et chaque classe de chiffres, la moitié des exemples disponibles est utilisée pour l'apprentissage, et l'autre moitié pour l'évaluation. Ceci représente 1295 chiffres pour l'apprentissage et 1294 chiffres pour l'évaluation.

Pour l'acquisition, on utilise le scanner "Helwett Packard SCANJET" piloté par le logiciel "SCANGAL" du PC COMPACQ 386. L'aquisition se fait en binaire à une résolution de 300 points par pouce. Après conversion du fichier fourni par le scanner en fichier image, et application de procédures de recherche et étiquetage des caractères, on calcule les différentes caractéristiques.

6.2 Résultats en multi-scripteurs inconnus

6.2.1 Comparaison des classifieurs

Sur la base d'évaluation de 14 scripteurs, le taux de reconnaissance obtenu est de 93.6% avec le réseau à connexions locales et apprentissage avec notion de voisinage

(réseau CL5). Des comparaisons ont été réalisées avec d'autres classifieurs: un réseau à 4 couches entièrement connecté [112+20+12+10 neurones + 3 neurones seuils] (réseau EC4) et les k-plus-proches-voisins (k-ppv). Le tableau suivant résume les résultats (les méthodes sont rangées par ordre de taux de généralisation décroissant).

Méthode	apprentissage	généralisation	nombre de multiplications
réseau CL5	**98.5%**	**93.6%**	**2439**
réseau EC4	99.4%	89.3%	2642
5-ppv	97.5%	88.7%	158368
ppv	100.0%	88.5%	158368
3-ppv	97.6%	88.2%	158368

Le temps d'apprentissage requis pour le réseau CL5 est de quelques heures sur Sun4. La matrice de confusion pour le réseau CL5 est fournie ci-dessous:

	C(0)	C(1)	C(2)	C(3)	C(4)	C(5)	C(6)	C(7)	C(8)	C(9)
C(0)	122	0	0	0	0	0	0	0	0	0
C(1)	1	114	0	1	2	0	2	0	1	1
C(2)	2	2	107	0	0	1	3	0	3	0
C(3)	0	0	0	109	0	0	0	0	2	5
C(4)	1	3	0	0	103	2	4	2	0	0
C(5)	0	0	0	1	0	113	1	0	3	0
C(6)	0	1	1	0	1	0	114	0	1	0
C(7)	0	6	0	1	2	0	0	105	1	2
C(8)	1	1	2	0	0	1	2	0	108	2
C(9)	0	1	0	5	0	0	0	1	0	105

6.2.2 Evaluation des caractéristiques

Afin de donner une indication quant à l'apport des diverses caractéristiques fournies au réseau, plusieurs expériences avec des sous-ensembles de caractéristiques ont été réalisées. Les résultats sont résumés ci-dessous. Tous les réseaux nommés EC sont entièrement connectés et comportent un neurone seuil sur chaque couche, sauf sur la couche de sortie. Pour ces réseaux, dans le cas de la prise en compte des profils uniquement, des résultats plus détaillés pourront être trouvés dans [*Accarie*90].

Caractéristiques statistiques:

Réseau EC2 : 2 couches (20+10 neurones + 1 seuil)

Réseau EC3 : 3 couches (20+15+10 neurones + 2 seuils).

Méthode	généralisation
3-ppv	83.5%
ppv	83.3%
réseau EC3	71.3%
réseau EC2	70.7%

Il est intéressant de noter que les plus proches voisins fournissent des résultats nettements meilleurs que les réseaux de neurones. Ceci est probablement dû à la nature de ces caractéristiques (chaque caractéristique apporte une information non négligeable indépendamment des autres, alors que pour les profils, c'est la forme générale qui est porteuse d'information).

Caractéristiques morphologiques:

Réseau EC2 : 2 couches (12+10 neurones + 1 neurone seuil)

Réseau EC3 : 3 couches (12+5+10 neurones + 2 neurones seuils)

Méthode	généralisation
réseau EC3	69.8%
ppv	66.3%
5-ppv	65.8%
3-ppv	65.8%
réseau EC2	62.3%

Malgré leur faible nombre, ces caractéristiques fournissent des résultats non négligeables, ce qui justifie les approches morphologiques proposées par plusieurs auteurs. Par contre, elles sont peu robustes: par exemple pour un zéro, le trou devient un ensemble de cavités dès que le chiffre est mal fermé, et il ne semble donc pas judicieux de construire un système sur la base de ces seules caractéristiques. Ceci explique éventuellement le fait que les plus proches voisins se comportent légèrement

moins bien que le réseau à 3 couches.

Profils gauche et droit:

Réseau EC2 : 2 couches (64+10 neurones + 1 neurone seuil)

Réseau EC3 : 3 couches (64+15+10 neurones + 2 neurones seuils)

Réseau EC4 : 4 couches (64+15+10+10 neurones + 3 neurones seuils)

Réseau CL5 : le réseau de la figure 7, sans les entrées profil orienté, caractéristiques morphologiques, et zones.

Méthode	généralisation
réseau CL5	90.1%
réseau EC4	87.8%
réseau EC3	87.1%
3-ppv	86.1%
ppv	86.0%
réseau EC2	85.5%

Profils gauche, droit et orienté:

Réseau EC2 : 2 couches (80+10 neurones + 1 neurone seuil)

Réseau EC3 : 3 couches (80+15+10 neurones + 2 neurones seuils)

Réseau EC4 : 4 couches (80+15+10+10 neurones + 3 neurones seuils)

Réseau CL5 : Le réseau de la figure 7 sans les entrées morphologiques et statistiques.

Méthode	généralisation
réseau CL5	92.3%
réseau EC3	90.9%
réseau EC4	88.3%
3-ppv	86.5%
ppv	86.4%
réseau EC2	86.0%

On remarquera ici l'apport du profil orienté qui permet de gagner 2.2% en performance (92.3% de reconnaissance contre 90.1% sans le profil orienté).

Profils gauche, droit, orienté et Zones:

Réseau EC2 : 2 couches (12+10 neurones + 1 neurone seuil)

Réseau EC3 : 3 couches (12+5+10 neurones + 2 neurones seuils)

Réseau CL5 : le réseau représenté figure 7, sans les 12 entrées morphologiques.

Méthode	généralisation
réseau CL5	93.1%
réseau EC3	89.9%
5-ppv	89.2%
ppv	88.5%
3-ppv	88.2%
réseau EC2	87.2%

La prise en compte des caractéristiques statistiques (zones) permet un gain de 0.8% en performances (93.1% de reconnaissance contre 92.3%). Si l'on utilise également les caractéristiques morphologiques, on gagne encore 0.5% (on atteint 93.6% de reconnaissance).

6.2.3 Visualisation des résultats

Un exemple de résultat est représenté figure 8. Sous chaque chiffre, le système a indiqué la classe trouvée, la vraie classe, et la confiance (écart relatif entre les 2 plus fortes sorties du réseau de neurones). Cette fiche appartient à l'ensemble d'évaluation. Ce résultat a été obtenu avec les profils uniquement et un réseau entièrement connecté de 3 couches, car l'interface permettant la visualisation des résultats obtenus avec le réseau à connexions locales n'est pas encore développée.

6.2.4 Résultats en fonction du seuil de rejet sur la confiance

Il est possible de réduire le taux d'erreur à condition d'accepter une non-décision (rejet) dans certains cas. Pour cela, on fixe un seuil sur la confiance. Lorsque la confiance obtenue est inférieure au seuil, il y a rejet. Le tableau suivant indique les variations des taux de rejet, de reconnaissance, et d'erreur, en fonction du seuil. Le réseau considéré est à 3 couches (80+15+10 neurones + 2 neurones seuils), et est alimenté par les profils gauche, droit, et orienté.

seuil	*rejet*	*reconnaissance*	*erreur*
0.0	0.0%	90.9%	9.1%
0.1	3.4%	89.1%	7.5%
0.2	7.1%	86.9%	6.0%
0.3	10.5%	84.9%	4.6%
0.4	13.4%	82.9%	3.7%
0.5	18.8%	79.1%	2.1%
0.6	23.7%	75.0%	1.3%
0.65	26.3%	72.7%	1.0%
0.7	30.3%	69.2%	0.5%
0.75	35.4%	64.3%	0.3%
0.8	42.2%	57.7%	0.1%
0.9	72.9%	27.0%	0.1%

6.3 Résultats en multi-scripteurs connus

Nous donnons ci-dessous les résultats obtenus lorsque le scripteur appartient à un groupe connu a priori (ici le groupe de 30 personnes). Les ensembles d'apprentissage et d'évaluation sont constitués de la façon suivante:

- Ensemble d'apprentissage : Pour chacun des 30 scripteurs, et pour chaque classe de chiffres, on prend la moitié des exemples disponibles. On a au total 1295 chiffres dans cet ensemble (les différentes classes et les différents scripteurs sont représentés par à peu près le même nombre d'échantillons).
- Ensemble d'évaluation : le complémentaire de l'ensemble d'apprentissage dans l'ensemble de tous les chiffres disponibles. On a au total 1294 chiffres dans cet ensemble.

Le tableau ci-dessous résume les résultats obtenus par différents réseaux ainsi que par les plus proches voisins:

- Réseau CL5 : le réseau de la figure 7
- Réseau EC2 : 2 couches, seuillé, et entièrement connecté (112+10 neurones + 1 neurone seuil)
- Réseau EC4 : 4 couches, seuillé, et entièrement connecté (112+20+12+10 neurones + 3 neurones seuils).

Méthode	apprentissage	généralisation
réseau CL5	99.0%	97.7%
3-ppv	97.8%	96.4%
réseau EC4	99.6%	96.4%
ppv	100%	96.2%
5-ppv	96.9%	96.1%
réseau EC2	96.5%	95.9%

6.4 Situation par rapport aux travaux antérieurs

Il est intéressant de comparer nos résultats avec [*LeCun*89] qui est considéré comme la référence actuelle, en gardant toutefois à l'esprit le fait que les bases de données sont différentes.

Au niveau des performances, Le Cun obtient 95% de reconnaissance et 5% d'erreur, ce qui est supérieur à nos résultats (93.6% de reconnaissance et 6.4% d'erreur). On peut toutefois espérer améliorer nos résultats en apprenant sur plus d'exemples, et surtout plus de scripteurs (nous avons appris sur 1414 exemples provenant de 16 scripteurs contre 7291 exemples provenant de 1458 scripteurs pour [*LeCun*89] !). D'un autre coté, la base de données utilisée par Le Cun est probablement plus difficile que la notre (il s'agit de chiffres extraits de codes postaux, fournis par la poste US). La comparaison des méthodes en terme de performances est donc pour l'instant délicate.

Au niveau de la complexité des méthodes, notre réseau comporte seulement 154 neurones et 2439 liaisons (il faudrait également tenir compte du prétraitement,

mais il est ici très simple), contre 1256 neurones et 64660 liaisons pour [*LeCun*89] (rappelons que le nombre de multiplications à effectuer pour une propagation est égal au nombre de liaisons). Cette simplicité peut être un avantage dans le cas où une réalisation matérielle serait envisagée.

A ce jour, il ne nous a pas été possible de faire converger le réseau de Le Cun sur notre base de données; sans savoir pour l'instant si cela est dû à un "bug" informatique, où à une particularité de la base de données (le réseau se trouve bloqué dans un minimum local, quelle que soit l'initialisation). Nous devons donc nous contenter d'indiquer, sur notre base de données, les performances de réseaux entièrement connectés alimentés par l'imagette 16x16 du chiffre:

Réseau	Apprentissage	Généralisation
2 couches (256+10)	78.2%	53.1%
4 couches (256+30+10+10)	94.7%	60.3%

7 Conclusion

Nous avons montré qu'une approche intermédiaire entre les méthodes classiques (qui travaillent sur un petit nombre de paramètres) et les méthodes "tout-neuronal" est possible.

Cette approche présente, par rapport aux méthodes classiques, l'avantage d'éviter les difficiles tâches de choix des paramètres, constitution des modèles, élaboration des règles de décision, etc.

Elle présente, par rapport aux approches "tout-neuronal", l'avantage de rester raisonnable en terme de complexité.

Au niveau des performances, il est difficile de conclure, car chaque auteur travaille sur sa base de données, ce qui rend toute comparaison délicate. Les performances obtenues avec notre approche semblent correctes, au vu des résultats annoncés dans la littérature. Enfin, on peut raisonnablement espérer des améliorations en apprenant sur des échantillons provenant d'un plus grand nombre de scripteurs.

Références

[*Accarie*90] Jean Paul ACCARIE
"Reconnaissance de chiffres manuscrits par réseaux de neurones"
Rapport de stage E.S.E., Septembre 1990
Thomson CSF/LER, Responsable de stage: G. BUREL

[*Beauville*90] J.P. ASSELIN de BEAUVILLE et al.
"Reconnaissance de chiffres manuscrits saisis sans contrainte"
BIGRE n^o68 (Reconnaissance automatique de l'écrit),
Le Havre, 18 mai 1990, pp 89-101

[*Gokana*86] Denis GOKANA
"Contribution à la reconnaissance automatique de caractères manuscrits.
Application à la lecture optique de caractères sur supports mobiles"
Thèse doctorat ès Sciences Physiques, Orsay, 28 Mars 1986

[*Govindan*90] V.K. GOVINDAN, A.P. SHIVAPRASAD
"Character Recognition - A Review"
Pattern Recognition, vol 23, n^o7, pp 671-683, 1990

[*LeCun*89] Y. Le Cun et al.
"Handwritten Digit Recognition :
Applications of Neural Network Chips and Automatic Learning"
IEEE Communications Magazine, November 1989

[*Mitchell*89] Brian T. MITCHELL, Andrew M. GILLIES
"A model based computer vision system
for recognizing handwritten ZIP codes"
Machine Vision and Applications, vol 2, n^o4, Fall 1989, pp 231-243

[*Pawlicki*88] Thaddeus F. PAWLICKI et al.
"Neural network models
and their application to handwritten digit recognition"
IEEE ICNN, San Diego, California, pp II.63-II.70, 24-27 July 1988

[*Ross*88] J. ROSS
"Growing Recognition"
Systems International, pp 79-80, September 1988

[*Shridhar*86] M. SHRIDHAR, A. BADRELDIN
"Recognition of isolated and simply connected handwritten numerals"
Pattern Recognition, vol 19, n^o1, pp 1-12, 1986

[*Shridhar*87] M. SHRIDHAR, A. BADRELDIN
"Context-directed segmentation algorithm for handwritten numeral strings"
Image and Vision Computing, vol 5, n^o1, February 1987

[*Stringa*89] L. STRINGA
"Efficient classification of totally unconstrained handwritten numerals with a trainable neural network"
Pattern Recognition Letters 10, pp 273-280, October 1989

[*Tappert*90] Charles C. TAPPERT, Ching Y. SUEN, Toru WAKAHARA
"The state of the art in On-Line handwriting recognition"
IEEE-PAMI, vol 12, n^o8, August 1990

Figure 8: *Quelques résultats*

Chapitre 9:

COMPRESSION D'IMAGES

La compression d'images est une tâche essentielle pour la sauvegarde et la transmission d'images. Comme la bande passante de la plupart des canaux de communication ou la capacité des mémoires de masse sont relativement inextensibles, la compression de données est nécessaire pour faire face à la quantité croissante d'information que l'on souhaite transmettre ou conserver. En conséquence, la théorie et la pratique de la compression d'images reçoivent actuellement une attention soutenue.

Nous nous intéresserons ici à la compression d'images statiques, c'est à dire que nous ne chercherons pas à exploiter les redondances entre images successives dans une séquence. Après un bref rappel des méthodes existantes pour la compression d'images statiques, nous proposerons une nouvelle méthode de compression par blocs d'images numériques. Cette méthode exploite les propriétés topologiques de l'algorithme de Quantification Vectorielle avec Voisinage (VQN). Cet algorithme a été présenté dans le chapitre 4, et nous avons établi des résultats théoriques qui permettent de mieux comprendre son fonctionnement. Divers résultats expérimentaux sont fournis, afin d'illustrer le principe et les avantages de la méthode proposée.

1 Généralités

1.1 Entropie d'une image

La compression d'image est possible car l'information contenue dans une image est très redondante. Par exemple, un grand nombre de zones uniformes sont généralement présentes sur une image. En conséquence, un pixel a toute les chances d'avoir une luminance proche de la luminance moyenne dans son voisinage, ce qui signifie que l'information qu'il apporte est faible.
Une mesure de l'information portée par une image est l'entropie:

$$H = -\sum_i p_i log_2 p_i$$

Où p_i est la probabilité de l'image i, et i parcourt l'ensemble de images possibles. Si toutes les images de N pixels et n niveaux de gris avaient la même probabilité, on aurait:

$$p_i = \frac{1}{n^N}$$

et en conséquence:

$$H = N log_2 n$$

Cette valeur correspond au volume de donnée classiquement alloué pour enregistrer une image numérique.

Mais, en fait les images ne sont pas équi-probables. Par exemple, il y a peu de chances pour qu'une image créée au hasard ressemble à une image réaliste. En conséquence, les images réalistes ne constituent qu'un très faible sous-ensemble de l'ensemble des images possibles, et l'entropie H est certainement bien plus faible que $N log_2 n$.

Une mesure expérimentale de H est impossible, car cela nécessiterait le recueil de statistiques sur l'ensemble des n^N images possibles. Toutefois, des mesures d'entropie sur des groupes de pixels voisins ont été fournies par W.F. SCHREIBER [*Schreiber*56]. Ces mesures furent obtenues sur des images à 64 niveaux de gris. Nous noterons u_{k-2}, u_{k-1}, u_k les luminances de pixels successsifs sur une ligne:

Entropie d'ordre 0

$$H_0 = -\sum_{l=0}^{63} P(u_k = l) log_2 P(u_k = l) = 4.4 bits$$

Entropie d'ordre 1

$$\begin{aligned} H_1 &= -\sum_{l_2} P(u_{k-1} = l_2) \sum_{l_1} P(u_k = l_1 \mid u_{k-1} = l_2) log_2 P(u_k = l_1 \mid u_{k-1} = l_2) \\ &= 1.9 bits \end{aligned}$$

C'est l'information moyenne apportée par le pixel k lorsque u_{k-1} est connu.

Entropie d'ordre 2

$$\begin{aligned} H_2 &= -\sum_{l_3} \sum_{l_2} P(u_{k-2} = l_3, u_{k-1} = l_2) \\ &\qquad \sum_{l_1} P(u_k = l_1 \mid u_{k-2} = l_3, u_{k-1} = l_2) log_2 P(u_k = l_1 \mid u_{k-2} = l_3, u_{k-1} = l_2) \\ &= 1.5 bits \end{aligned}$$

Ceci est l'information moyenne apportée par le pixel k lorsque les luminances u_{k-1} et u_{k-2} sont connues.

En conclusion, des pixels voisins sont fortements corrélés car la connaissance du voisinage cause une forte décroissance de l'entropie. C'est cette redondance que les techniques de compression visent à exploiter.

1.2 Techniques existantes

Les techniques de compression dédiées aux images statiques peuvent être groupées en quatre grandes familles, sur la base de leur nature spécifique:

1. **Les techniques prédictives** exploitent la redondance de l'image pour prédire la luminance d'un pixel en fonction de son voisinage.
2. **La quantification vectorielle** exploite la redondance de l'image pour définir un dictionnaire de blocs image standards.
3. **Les techniques par transformées** réalisent la compression de blocs d'images par des transformations à conservation d'énergie, qui concentrent le maximum d'information sur un nombre minimal d'échantillons.
4. **Les techniques en sous-bande** assument une certaine indépendance des différentes bandes de fréquence spatiale pour les coder séparément.

Pour une revue des techniques de compression, on pourra consulter [*Jain*81] ou [*Forchheimer*89]. Les techniques prédictives sont les plus simples à implémenter, mais sont généralement les moins robustes, et ne permettent pas d'atteindre des taux de compression élevés (généralement, elles exploitent seulement les redondances entre pixels adjacents). Les méthodes par transformées, ou par quantification vectorielle sont préférées lorsque des taux de compression importants doivent être atteints. Leurs meilleures performances sont dûes au fait qu'elles prennent en compte des blocs d'image assez grands, et exploitent donc une plus grande variété de redondances.

Une étude comparative partielle est proposée dans [*Fischer*89]. Elle montre un léger avantage des techniques en sous-bande sur les techniques par transformées lorsque le débit est faible (moins de 1 bit/pixel). Si l'on remplace la quantification scalaire traditionnellement employée avec ces 2 types de techniques par une quantification vectorielle on obtient un gain important (de 2 à 5 dB sur le rapport signal à bruit), et l'écart entre les techniques par tranformée et les techniques en sous-bande se restreint.

Dans le domaine des réseaux de neurones, Cotrell, Munro, et Zipser [*Cottrel*87] ont proposé une méthode de compression d'images qui entre dans la catégorie des méthodes par transformée. Cette méthode est basée sur un perceptron à 3 couches, qui réalise "l'identity mapping" de blocs d'images via une couche intermédiaire de faible taille. Elle s'apparente donc aux méthodes par transformée. Les expériences que nous avons menées avec cette méthode montrent toutefois qu'elle n'est intéressante que sur des images simples, ou lorsque le débit est important (plus de 2 bits/pixel).

Nous proposons une méthode qui entre dans la catégorie des méthodes par quantification vectorielle. Nous utilisons un algorithme qui s'inspire du modèle des "cartes topologiques" de Kohonen. Une première version de cette méthode a été présentée dans [*Burel*89], et son implémentation sur une machine parallèle à base de transputers a été réalisée et validée [*Burel*91].

Dans le paragraphe 2, nous dirons quelques mots des images utilisées pour les expérimentations, et nous comparerons différents algorithmes capables de créer un dictionnaire pour la Quantification Vectorielle. Dans le paragraphe 3, nous présenterons la méthode proposée.

2 Expérimentations de Quantification Vectorielle

2.1 Principe de la compression d'image par Quantification Vectorielle

La Quantification Vectorielle est une technique de compression qui a été très largement utilisée ([*Goldberg*86], [*Aravind*87], [*Fischer*89] par exemple). La méthode consiste à découper l'image en blocs et à remplacer chaque bloc par un index. Cet index se rapporte à un dictionnaire contenant un nombre limité de blocs représentatifs ("prototypes"). La constitution du dictionnaire est habituellement réalisée par l'algorithme des "k-means" ou l'algorithme LBG [*Linde*80].

A la reconstruction, on remplace l'index par le prototype correspondant.

En notant K le nombre de pixels dans un bloc et M le nombre d'éléments du dictionnaire, le taux de compression obtenu est:

$$T_c = \frac{log_2 M}{K} \; bits/pixel$$

Pour maintenir une faible distorsion sur une grande variété d'images, un dictionnaire de taille importante est nécessaire. De plus, pour obtenir un codage efficace, on doit prendre en compte des blocs de taille importante (afin d'exploiter au mieux les redondances). Ceci pose d'importantes difficultés en terme de puissance de calcul requise.

C'est pourquoi, nous préférons une approche dans laquelle le dictionnaire est adapté à chaque image (ou à un groupe d'images dans le cas où l'on a des ensembles d'images assez semblables). Un tel point de vue a déjà été suggéré dans [*Goldberg*86]. Ceci nous permet d'avoir un dictionnaire de taille restreinte, et de prendre des blocs d'image de petite taille. Pour nos expérimentations, nous prendrons des blocs de 3x3 pixels et 256 prototypes. On a donc:

$$\begin{aligned} K &= 9 \\ M &= 256 \\ T_c &= 0.89 \; bits/pixel \end{aligned}$$

La contrepartie est la nécessité de transmettre le dictionnaire avec chaque image (ou groupe d'images). Mais nous verrons plus loin que ceci n'est pas très coûteux. D'autre part, afin de réduire le temps de calcul nécessaire à la constitution du dictionnaire, on initialisera les prototypes avec le contenu d'un dictionnaire standard

(constitué à partir de plusieurs images). De plus, les algorithmes de constitution du dictionnaire sont massivement parallèles, d'où la possibilité d'une implémentation matérielle efficace.

2.2 La base d'images utilisée pour les expérimentations

Pour les expérimentations, nous utiliserons les 4 images présentées sur les figures 8 (foot), 9 (Kiel), 10 (mobile) et 11 (bateau). Ces images servent souvent de références dans le domaine de la compression d'images du fait de leur difficulté intrinsèque. Il est par exemple très difficile de comprimer l'image "Kiel" sans dégrader les infimes détails quelle présente (filins, détails de l'arrière plan, ...). De même, l'image "mobile" présente des contours très tranchés. Toute dégradation de ces contours se remarquera facilement, et laissera une impression désagréable à l'œil. Elle présente également des zones quasi-uniformes qui sont bien connues pour être "redoutables" vis-à-vis des méthodes de compression par blocs, car on apperçoit souvent les blocs sur ces zones.

Ces images sont codées sur 256 niveaux de gris (8 bits/pixel). Afin de simplifier les calculs relatifs à la suite de l'exposé, on travaillera sur des sous-images dont la taille est un multiple de 9.

image	entropie	taille	taille sous-image	nb d'octets	nb de blocs 3x3
foot	6.92	536x676	531x675	358425	39825
Kiel	7.26	576x720	576x720	414720	46080
mobile	7.63	536x676	531x675	358425	39825
bateau	6.96	512x512	504x504	254016	28224

2.3 Comparaison des algorithmes de Quantification Vectorielle

2.3.1 Objectifs

Notre objectif dans cette section est de comparer les différents algorithmes capables de réaliser la Quantification Vectorielle de blocs d'images. Les algorithmes comparés sont les suivants:

- L'algorithme des "k-means" (k-means)
- L'algorithme des "k-means" avec initialisation par "splitting" (LBG)
- L'algorithme de Kohonen (KH)
- Les algorithmes de Quantification Vectorielle avec Voisinage (VQN et VQNf) proposés au chapitre 4.

On travaille sur des blocs 3x3 et le nombre de neurones (ou prototypes) est de 256. L'apprentissage se fait sur un ensemble de blocs extraits des images "foot", "Kiel", et "mobile". On prend un bloc sur 8, soit un total de 15716 blocs. Le nombre d'itérations accordé est T=200.

Pour les algorithmes possédant une notion de voisinage, les coefficients déterminant l'influence du voisinage sont:

$$\alpha_{jk} = \alpha_0 e^{-\frac{(j-k)^2}{2\sigma_t^2}}$$

L'écart type σ_t décroît avec le nombre d'itérations (t) suivant une loi exponentielle:

$$\sigma_t = \sigma_0 \left(\frac{\sigma_{T-1}}{\sigma_0}\right)^{\frac{t}{T-1}}$$

Les valeurs initiales et finales de l'écart type sont:

$$\begin{aligned} \sigma_0 &= 4.0 \\ \sigma_{T-1} &= 0.2 \end{aligned}$$

Enfin, l'initialisation des poids des neurones (ou prototypes) est réalisée de la façon suivante (sauf pour LBG qui possède sa propre initialisation):

1. Les poids correspondant à un neurone (ou prototype) sont choisis égaux à son numéro.
2. On bruite ces poids avec un bruit additif de densité de probabilité uniforme entre -5.0 et $+5.0$.

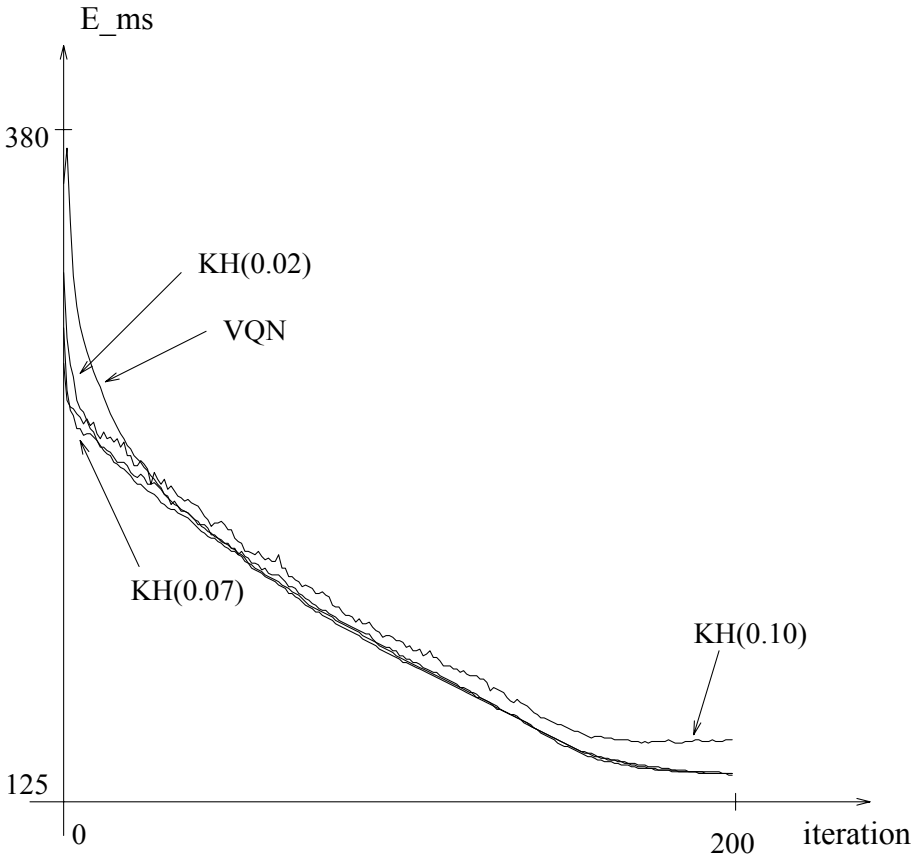


Figure 1: Influence de la vitesse d'apprentissage (Kohonen)

2.3.2 Choix de la vitesse d'apprentissage pour l'algorithme de Kohonen

la figure 1 montre l'évolution de l'erreur quadratique moyenne suivant la valeur de la vitesse d'apprentissage α_0. On constatera qu'une vitesse d'apprentissage importante (0.10) conduit à une convergence moins bonne.

Pour des valeurs correctes de la vitesse d'apprentissage, la courbe d'apprentissage de l'algorithme de Kohonen est très proche de celle de l'algorithme VQN.

2.3.3 Comparaison des algorithmes VQN, VQNf, LBG et k-means

La figure 2 permet de comparer les courbes d'apprentissage de 4 algorithmes. Il est clair que l'algorithme des k-means a été bloqué dans un minimum local peu

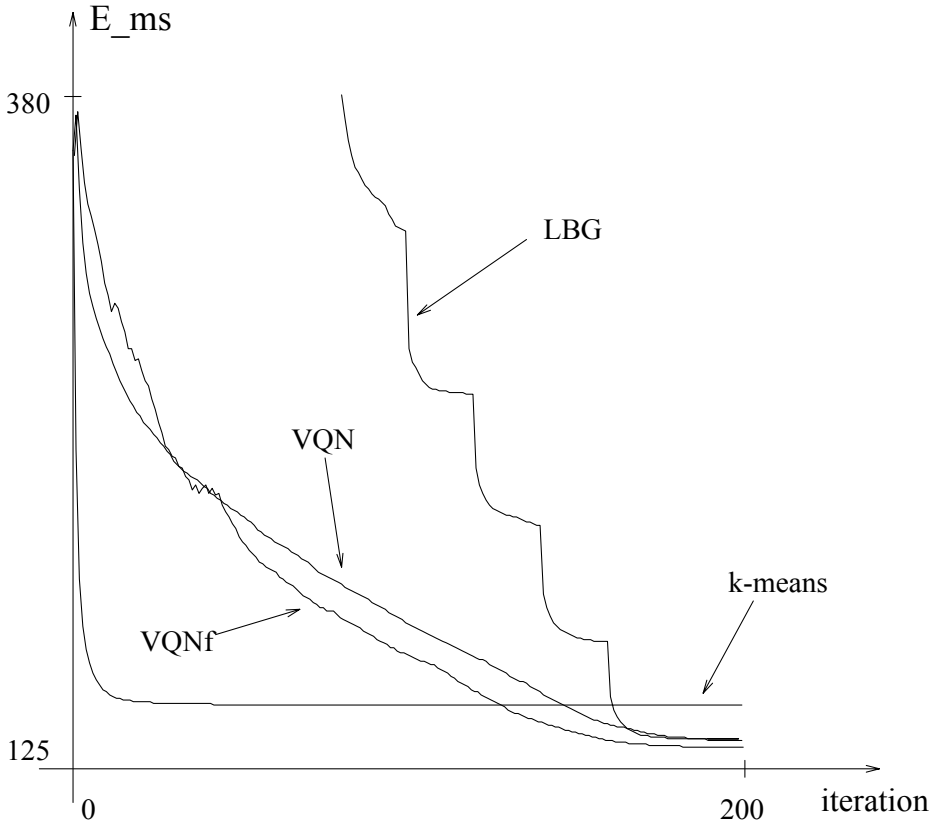


Figure 2: Comparaison des algorithmes VQN, VQNf, LBG et k-means

satisfaisant. L'initialisation par "splitting" proposée dans [*Linde*80] pour réduire ce risque semble efficace (LBG).

Les algorithmes LBG, VQN et VQNf conduisent à des solutions très proches. Les prototypes trouvés par l'algorithme VQN sont représentés sur la figure 3. Cette image est constituée des blocs prototypes (3x3 pixels), disposés dans le sens de la lecture. Il est clair que les prototypes représentent les configurations les plus courantes dans un bloc 3x3 (zones uniformes, contours de différentes orientations, etc). L'algorithme VQN a conservé la topologie (des prototypes voisins ont un contenu voisin).

Le tableau suivant indique les résultats obtenus (Erreur Quadratique Moyenne) lorsque l'on comprime les images avec le jeu de prototypes constitué par VQN. On rappelle que l'image "bateau" ne faisait pas partie de la base d'apprentissage. On

Figure 3: Prototypes (VQN)

peut également définir un rapport signal à bruit:

$$RSB = 10\ log\left(\frac{255^2}{e_{QM}}\right)$$

image	EQM	RSB
foot	51.3	31.0dB
Kiel	203.2	25.1dB
mobile	204.8	25.0dB
bateau	52.9	30.9dB

2.4 Apprentissage adapté à l'image

Nous avons justifié précédemment l'approche qui consiste à adapter le dictionnaire à l'image. L'apprentissage se fait sur l'image à coder, avec l'algorithme VQN. Le réseau est initialisé avec les poids obtenus précédemment (apprentissage global sur 3 images). On apprend pendant $T = 25$ itérations ($\sigma_0 = 1.0,\ \sigma_{T-1} = 0.2$, décroissance exponentielle) sur 1/3 des blocs de l'image.

Le tableau suivant indique les résultats obtenus:

image	EQM	RSB
foot	32.58	33.0 dB
Kiel	175.85	25.7 dB
mobile	168.47	25.9 dB
bateau	27.44	33.7 dB

On doit alors transmettre le dictionnaire en supplément des index. Le tableau suivant indique les taux de compression pour chacune des images:

image	index	prototypes	taux de compression
foot	39825 octets	2304 octets	0.94 bpp
Kiel	46080 octets	2304 octets	0.93 bpp
mobile	39825 octets	2304 octets	0.94 bpp
bateau	28224 octets	2304 octets	0.96 bpp

3 Exploitation de la topologie

3.1 Idée générale

Nous avons vu que l'algorithme VQN fournit de bons résultats tout en préservant la topologie, et sans nécessiter de réglage de paramètres. C'est l'algorithme que nous utiliserons pour la suite de l'étude.

Lorsque l'on comprime une image par blocs, comme nous l'avons fait à la section précédente, on n'exploite pas les redondances entre blocs voisins. Ainsi, par exemple, le bloc situé immédiatement à droite d'un bloc contenant un contour horizontal a toutes les chances de contenir le prolongement de ce contour. On peut donc envisager une amélioration qui consisterait à prédire l'index correspondant à un bloc à partir des index des blocs voisins.

Toutefois, pour que cette prédiction soit intéressante, il faut que les prototypes respectent une certaine topologie. Ainsi, l'histogramme de l'erreur de prédiction sera centré autour du zéro, et l'entropie de cette erreur sera faible. Si, par contre, les prototypes ne respectent aucune topologie, l'histogramme de l'erreur de prédiction sera plat, et l'entropie élevée (on ne peut alors espérer aucun gain).

Pour illustrer ce point, supposons que les blocs A et B soient voisins dans l'image. L'index de A est 135, et correspond à un contour horizontal. Dans ce cas, on peut prédire que l'index de B sera également 135. En pratique, il y a peu de chances pour que le contenu de B soit exactement le même que celui de A. Mais il en est probablement proche. Supposons, par exemple, que dans B le contour se soit légèrement incurvé, et examinons l'intérêt d'une conservation de la topologie:

- Si les prototypes respectent la topologie, l'index réel de B sera proche de 135 (par exemple 137) car le contenu de B est proche du contenu de A. L'erreur de prédiction sera faible (ici $137 - 135 = 2$).

- Si les prototypes ne respectent pas la topologie, le code réel de B n'a aucune raison d'être proche de celui de A (il sera par exemple 212) et l'erreur de prédiction sera souvent élevée (ici $212 - 135 = 77$).

Figure 4: image des index de "foot" (VQN)

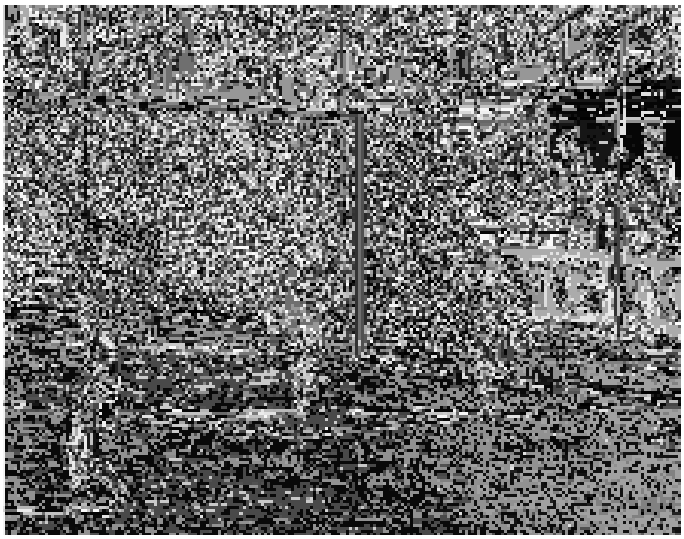


Figure 5: image des index de "foot" (LBG)

Lorsque l'on comprime une image, on obtient une nouvelle image (de taille réduite) dans laquelle chaque bloc de 3x3 pixels est remplacé par l'index correspondant à ce bloc. Cette image sera à présent nommée "image des index". La figure 4 représente l'image des index de "foot" obtenue avec l'algorithme VQN. Cette image possède une cohérence interne qui est dûe à la conservation de la topologie. Ce n'est pas le cas de la figure 5 qui représente les index de "foot" obtenus avec l'algorithme LBG.

L'idée consiste donc à comprimer à nouveau l'image des index, car elle possède des redondances internes. Pour réaliser cette compression, on peut utiliser un prédicteur, ou bien réaliser une nouvelle quantification vectorielle sur cette image.

3.2 Méthode proposée

Nous proposons un procédé à 2 étages pour réaliser la quantification vectorielle d'images. Les différences par rapport à la technique classique sont les suivantes:

1. Le dictionnaire est créé par un algorithme topologique (par exemple VQN ou VQNf).
2. Les index sont à nouveau comprimés grâce à un prédicteur constitué d'un perceptron multicouches à Représentation Scalaire Distribuée. Les erreurs de prédiction sont codées en Huffman.

Les avantages sont les suivants:

1. Les algorithmes topologiques sont moins vulnérables au piégeage dans un minimum local que les "k-means".
2. La conservation de la topologie permet la mise en œuvre d'un second étage (prédicteur).
3. La compression des index permet d'améliorer le taux de compression sans perte de qualité.

3.3 Le codeur et le décodeur

On décrit ci-dessous le fonctionnement du codeur et du décodeur (fig 6) en supposant que le dictionnaire, les coefficients du prédicteur, et les codes de Huffman ont déjà été élaborés.

On utilise les notations suivantes:

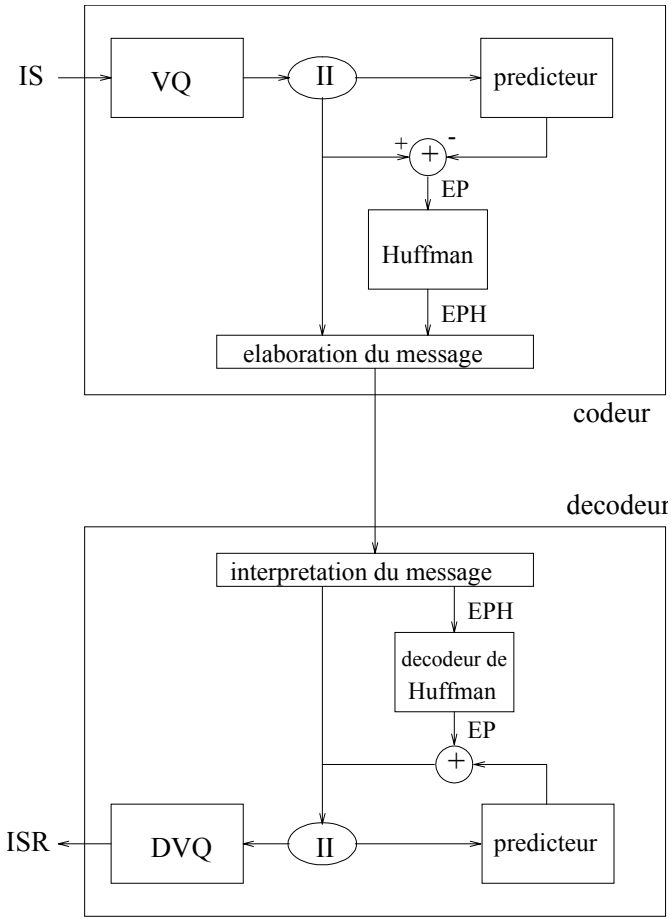


Figure 6: Compression à 2 niveaux (VQ + prédicteur)

IS : Image Source
ISR : Image Source Reconstruite
II : Image des Index
EP : Erreurs de Prédiction
EPH : Erreurs de Prédiction codées en Huffman
VQ : Quantificateur Vectoriel
DVQ : Décodeur Vectoriel
P : Indicateur de mise en œuvre du prédicteur
RD : Indicateur de remise à jour du dictionnaire
RP : Indicateur de remise à jour des coefficients du prédicteur
RH : Indicateur de remise à jour des codes de Huffman

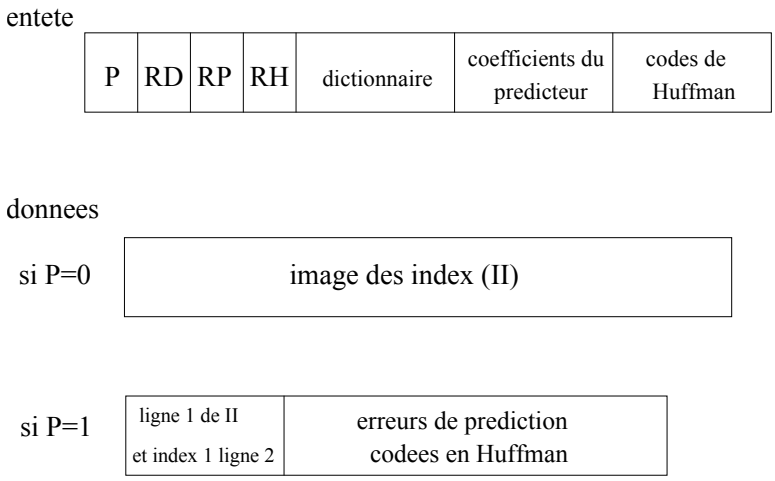


Figure 7: Exemple de format pour l'image codée

3.3.1 Le codeur

Le codeur reçoit en entrée l'image source (IS), et lui fait subir les traitements suivants (fig 6):

Etage de Quantification Vectorielle (VQ)

L'image source est découpée en blocs et chaque bloc est comparé aux éléments du dictionnaire, en vue de le remplacer par l'index du prototype qui s'en rapproche le plus. Si on note $\vec{W}_j$ le prototype d'index j dans le dictionnaire, et $\vec{x}$ le bloc image à coder, $\vec{x}$ sera remplacé par l'index k tel que:

$$\forall j \neq k \qquad \|\vec{x} - \vec{W}_k\|^2 \leq \|\vec{x} - \vec{W}_j\|^2$$

Lorsque tous les blocs de l'image source ont été remplacés par un index, on obtient l'image des index (II). Par convention, les index sont des entiers compris entre 0 et $M-1$, où M est la taille du dictionnaire.

Etage de prédiction

Le prédicteur est un réseau de neurones à Représentation Scalaire Distribuée (chap. 4). Pour prédire la valeur de l'index $i(x,y)$ situé à la colonne x et à la ligne y dans II, il reçoit en entrée un voisinage causal (par exemple $i(x-1,y-1)$, $i(x,y-1)$, $i(x+1,y-1)$ et $i(x-1,y)$). Nous noterons $\hat{\imath}(x,y)$ la prédiction qu'il fournit.

Afin de ne pas induire de perte de qualité, les erreurs de prédiction $i(x,y)-\hat{i}(x,y)$ sont transmises au décodeur. Pour cela, on les représente modulo M, et on les code en longueur variable (code de Huffman).

Elaboration du message

Afin de faciliter la compréhension du procédé, on donne figure 7 un format possible pour le flux de données entre le codeur et le décodeur. Il est clair que l'originalité de la méthode n'est pas dans ce format particulier, mais bien dans la structure à 2 étages, et dans les algorithmes utilisés pour la prédiction (RSD) et pour la constitution du dictionnaire (VQN ou VQNf).

Le message est constitué d'un entête et des données proprement dites.

- Le premier bit du message (P) indique si le prédicteur est utilisé ou non. Il peut en effet arriver, dans certains cas "pathologiques", que la mise en œuvre du prédicteur n'apporte aucun gain en termes de compression. Il est alors préférable de transmettre directement l'image des index.

- Les 3 bits suivants (RD, RP, RH) indiquent si l'on doit remettre à jour le dictionnaire, les coefficients du prédicteur, ou les codes de Huffman. Dans le cas d'une sauvegarde ou d'une transmission d'une séquence d'images, de telles remises à jour seront effectuées lorsque l'image source courante est très différente des images précédentes (i.e. e_{QM} devient élevé).

On ajoute ensuite, s'il y a lieu, le dictionnaire, les coefficients du prédicteur, et/ou les codes de Huffman.
Les 4 premiers octets du champ "prédicteur" indiquent, en virgule flottante, un coefficient de normalisation γ. Les coefficients du prédicteur sont ensuite rangés sous forme d'entiers (2 octets par coefficient). Si C_k est l'entier représentant le coefficient numéro k, sa vraie valeur est γC_k.
Le premier octet du champ "codes de Huffman" contient l'entier immédiatement supérieur à $log_2 L_{max}$, où L_{max} est la longueur maximale d'un code de Huffman. Nous nommerons θ cet entier. Puis, chaque code est rangé sous la forme:

Longueur du code (L_k) {sur θ bits}	code {sur L_k bits}

Ensuite, si $P = 0$, on ajoute l'image des index. Si $P = 1$, on ajoute la première ligne de l'image des index, ainsi que le premier index de la 2^e ligne. Ceci permet d'initialiser le prédicteur. On a ensuite les erreurs de prédiction codées en Huffman.

3.3.2 Le décodeur

Le décodeur reçoit le message élaboré par le codeur. S'il y a lieu, il remet à jour le dictionnaire, les coefficients du prédicteur, et/ou les codes de Huffman. Si $P = 0$, il récupère l'image des index. Si $P = 1$, il récupère le début de l'image des index et transmet les erreurs de prédiction au décodeur de Huffman. Un prédicteur identique à celui du codeur permet de reconstruire progressivement l'image des index. Il s'agit d'une reconstruction exacte car on connait exactement les erreurs de prédiction.

Lorsque l'image des index est reconstruite, le décodeur vectoriel (DVQ) remplace chaque index j par l'élément $\vec{W}_j$ du dictionnaire. Le résultat est l'image source reconstruite (ISR).

3.4 Résultats expérimentaux (VQN + Prédiction)

3.4.1 Comparaison de différents prédicteurs

On compare différents prédicteurs neuronaux (perceptrons multicouches). Les neurones des couches cachées possèdent une fonction de transition en tangente hyperbolique. Les connexions sont totales entre 2 couches successives. Le neurone de sortie est linéaire.

Pour les réseaux classiques, on a un neurone seuil sur chaque couche (sauf en sortie). Pour les réseaux à représentation scalaire distribuée, on a (cf notations chap. 4):

$$h(u) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{u^2}{2\sigma^2}}$$

et

$$\begin{aligned} F(u) &= \tfrac{(\mu-1)}{2}(th(u)+1) \\ \sigma &= 2.5 \\ n_1 &= 0 \\ n_2 &= \mu - 1 \end{aligned}$$

On utilise les symboles suivants:

R2 : réseau classique 4+1 neurones (5 poids)
R3 : réseau classique 4+8+1 neurones (49 poids)
R4 : réseau classique 4+12+6+1 neurones (145 poids)
RSD5 : réseau à codage distribué 4+2+1 neurones, $\mu = 5$ (50 poids)
RSD8 : réseau à codage distribué 4+2+1 neurones, $\mu = 8$ (80 poids)
RSD16 : réseau à codage distribué 4+2+1 neurones, $\mu = 16$ (160 poids)

L'apprentissage se fait sur l'image des index. Pour prédire l'index $i(x, y)$, le prédicteur reçoit en entrée les 4 index constituant le voisinage causal adjacent: $i(x-1, y-1)$, $i(x, y-1)$, $i(x+1, y-1)$ et $i(x-1, y)$ Les tableaux suivants résument les résultats obtenus. EQM désigne la moitié de l'erreur quadratique moyenne de prédiction (sans mise en œuvre d'un limiteur en sortie du prédicteur). La prédiction est ensuite arrondie à l'entier le plus proche et limitée à l'intervalle $[0, M-1]$. Puis, les erreurs de prédiction sont représentées modulo M, et leur entropie est indiquée dans la colonne 3. *Lmoy* indique la longueur moyenne par index des codes de Huffman, *Lmax* la longueur maximale d'un code de Huffman, et $\sum l_k$ la somme des longueurs des codes (en bits).

Image "foot":

réseau	EQM	entropie de l'erreur	*Lmoy*	*Lmax*	$\sum l_k$
R2	386.6	6.52	6.55	14	2506
R3	382.1	6.53	6.55	14	2510
R4	380.3	6.49	6.52	14	2506
RSD8	370.3	6.49	6.52	14	2526
RSD16	297.5	6.37	**6.39**	15	2555

Image "Kiel":

réseau	EQM	entropie de l'erreur	*Lmoy*	*Lmax*	$\sum l_k$
R2	484.3	6.63	6.65	13	2436
R3	483.0	6.62	6.65	13	2445
R4	481.9	6.62	6.65	13	2445
RSD8	469.9	6.59	6.62	14	2471
RSD16	426.6	6.51	**6.53**	14	2490

Image "mobile":

réseau	EQM	entropie de l'erreur	*Lmoy*	*Lmax*	$\sum l_k$
R2	637.0	6.60	6.62	12	2366
R3	635.2	6.77	6.79	12	2378
R4	633.2	6.71	6.73	12	2370
RSD8	604.9	6.52	6.56	13	2408
RSD16	596.0	6.40	**6.45**	13	2412

Image "bateau":

réseau	EQM	entropie de l'erreur	*Lmoy*	*Lmax*	$\sum l_k$
R2	338.2	5.90	5.92	15	2580
R3	332.9	5.89	5.91	14	2572
R4	326.3	5.92	5.94	15	2608
RSD5	317.8	5.78	5.81	15	2595
RSD8	309.3	5.59	5.63	15	2612
RSD16	294.1	5.50	**5.55**	15	2604

Poids des prédicteurs linéaires (R2):

A titre d'information, on indique ci-dessous les poids des prédicteurs linéaires (réseaux R2). Les poids sont indiqués conformément à la disposition du voisinage causal pris en compte. Le nombre situé à gauche correspond au poids affectant la connexion avec le neurone seuil.

foot:

0.034	−0.190	0.460	0.068
	0.588		

Kiel:

0.049	−0.149	0.460	0.070
	0.516		

mobile:

0.019	−0.049	0.385	0.150
	0.471		

bateau:

0.023	−0.194	0.495	0.139
	0.521		

3.5 Taux de compression obtenus

On donne ci-dessous les taux de compression obtenus avec et sans la mise en œuvre du prédicteur.

- Le dictionnaire occupe 2304 octets.
- Le prédicteur pris en compte est RSD16. Avec le format proposé, ses coefficients occupent 324 octets (160x2+4).
- La longueur maximale des codes de Huffman est 15 (d'où $\theta = 4bits$). La place occupée par le champ "codes de Huffman" est donc 8+4x256+$\sum l_k$ bits, soit $129 + \frac{\sum l_k}{8}$ octets.

La taille de l'entête est donc de $2757 + \frac{\sum l_k}{8}$ octets.

Taille des images comprimées (en octets):

image	entête	données	total	rappel taille image
foot	3077	31947	35024	358425
Kiel	3069	37746	40815	414720
mobile	3059	32240	35299	358425
bateau	3083	20086	23169	254016

Gain en taux de compression dû au prédicteur:

image	T_c sans prédicteur	T_c avec prédicteur	gain
foot	0.94 bpp	0.78 bpp	17%
Kiel	0.93 bpp	0.79 bpp	15%
mobile	0.94 bpp	0.79 bpp	16%
bateau	0.96 bpp	0.73 bpp	24%

3.6 Variante (VQ+VQ)

Une variante possible à la méthode proposée consiste à comprimer l'image des index par une nouvelle quantification vectorielle. L'image des index est comprimée par blocs 3x3 et le dictionnaire du niveau 2 contient 256 prototypes. L'algorithme d'apprentissage est toujours VQN, et l'apprentissage est adapté à l'image des index du niveau 1. Les différences sont codées en Huffman et transmises au décodeur. Il est également nécessaire de transmettre le dictionnaire du niveau 2, et l'image des index du niveau 2. En conséquence, bien que l'entropie des différences soit plus faible que l'entropie des erreurs de prédiction, cette variante n'est pas plus intéressante que la méthode décrite précédemment. Les résultats ci-dessous ne sont donc donnés qu'à titre d'information.

image	EQM (niveau 2)	entropie des différences	$Lmoy$	$Lmax$	$\sum l_k$
foot	252.34	5.66	5.69	16	1841
Kiel	297.00	5.73	5.76	15	2180
mobile	322.27	5.61	5.64	15	2267
bateau	161.70	4.80	4.82	15	1612

- Le dictionnaire du niveau 1 occupe 2304 octets.
- Le dictionnaire du niveau 2 occupe 2304 octets.
- Pour les images "Kiel", "mobile", et "bateau", la longueur maximale des codes de Huffman est 15 (d'où $\theta = 4$ *bits*). La place occupée par le champ "codes de Huffman" est donc 8+4x256+$\sum l_k$ bits, soit $129 + \frac{\sum l_k}{8}$ octets. Pour l'images "foot", la longueur maximale des codes de Huffman est 16, d'où $\theta = 5$ *bits*. La

place occupée par le champ "codes de Huffman" est alors 8+5x256+$\sum l_k$ bits, soit $161 + \frac{\sum l_k}{8}$ octets.

La taille de l'entête est donc de $4770 + \frac{\sum l_k}{8}$ octets pour l'image "foot", et de $4738 + \frac{\sum l_k}{8}$ octets pour les autres images.

Taille des images comprimées (en octets):

image	entête	index niveau 2	différences	total	rappel taille image
foot	5001	4425	28326	37752	358425
Kiel	5011	5120	33178	43309	414720
mobile	5022	4425	28077	37524	358425
bateau	4940	3136	17005	25081	254016

Gain en taux de compression dû au second étage:

image	T_c sans 2^e étage	T_c avec 2^e étage	gain
foot	0.94 bpp	0.84 bpp	11%
Kiel	0.93 bpp	0.83 bpp	11%
mobile	0.94 bpp	0.84 bpp	11%
bateau	0.96 bpp	0.79 bpp	18%

4 Conclusion

Nous avons montré comment il est possible d'exploiter les propriétés topologiques d'algorithmes dérivés de l'algorithme de Kohonen dans le domaine de la compression d'images. On peut tirer de cette étude les conclusions suivantes:

- Les algorithmes VQN et VQNf sont capables de créer un dictionnaire aussi performant (en terme d'erreur quadratique) que l'algorithme LBG (et même légèrement meilleur pour VQNf). On a vu au chap. 4 que lorsque la dimension de l'entrée se restreint (c.a.d. lorsque que la densité de minima locaux augmente), l'avantage de VQNf sur LBG s'accroit.

- Contrairement à LBG, ces algorithmes préservent la topologie, ce qui permet de maintenir la cohérence de l'image des index, et autorise une nouvelle compression de cette image.
- Les réseaux à codage distribué (RSD) se révèlent être des prédicteurs efficaces. L'avantage par rapport aux prédicteurs linéaires, et par rapport aux perceptrons multicouches classiques est non-négligeable.

A qualité d'image égale, le gain sur le taux de compression dû à la possibilité de mise en œuvre d'un prédicteur est de 15% à 25%. Ces taux correspondent à la base d'images utilisée. Ces images étant plus difficiles que la moyenne, on peut espérer un gain plus important sur des images plus classiques (visages,...).

Références

[*Aravind*87] R. ARAVIND, Allen GERSHO
"Image compression based on Vector Quantization with finite memory"
Optical Engineering, July 1987, vol 26, n^o7

[*Burel*89] Gilles BUREL, Isabelle POTTIER
"Procédé de compression d'images
par auto-organisation d'un réseau neuronal"
Brevet n^o8916111, 6 décembre 1989

[*Burel*91] Gilles BUREL, Isabelle POTTIER
"Vector Quantization of Images using Kohonen algorithm
-Theory and Implementation- "
Revue Technique Thomson CSF, vol 23, n^o1, mars 1991

[*Cottrel*87] G.W.COTTREL, P.MUNRO, D.ZIPSER
"Image compression by backpropagation"
ICS Report 8702, February 1987, University of California, San Diego

[*Fischer*89] Thomas R. FISCHER, Mary E. BLAIN
"A comparison of Vector Quantization Subband and Tranform Coding of Imagery"
International Symposium on Circuits and Systems, 1989, pp 1520-1523

[*Forchheimer*89] Robert FORCHHEIMER, Torbjörn KRONANDER
"Image coding - From waveforms to animation"
IEEE trans. on ASSP, vol 37, n^o12, December 1989

[*Goldberg*86] Morris GOLDBERG, Paul R. BOUCHER
"Image compression using adaptative Vector Quantization"
IEEE trans. on Communications, vol COM-34, n^o2, February 1986

[*Jain*81] A.K.JAIN
"Image Data Compression : A Review"
Proceedings of the IEEE, vol 69, $n°$3, March 1981

[*Linde*80] Y.LINDE, A.BUZO, R.M.GRAY
"An algorithmm for Vector Quantizer design"
IEEE Trans. on Communications
vol COM-28, $n°$1, January 1980

[*Schreiber*56] W.F.SCHREIBER
"The measurement of third order probability distributions of television signals"
IRE Trans. Inform. Theory
vol IT-2, pp. 94-105, September 1956

Figure 8: Image "foot" et sa reconstruction après compression

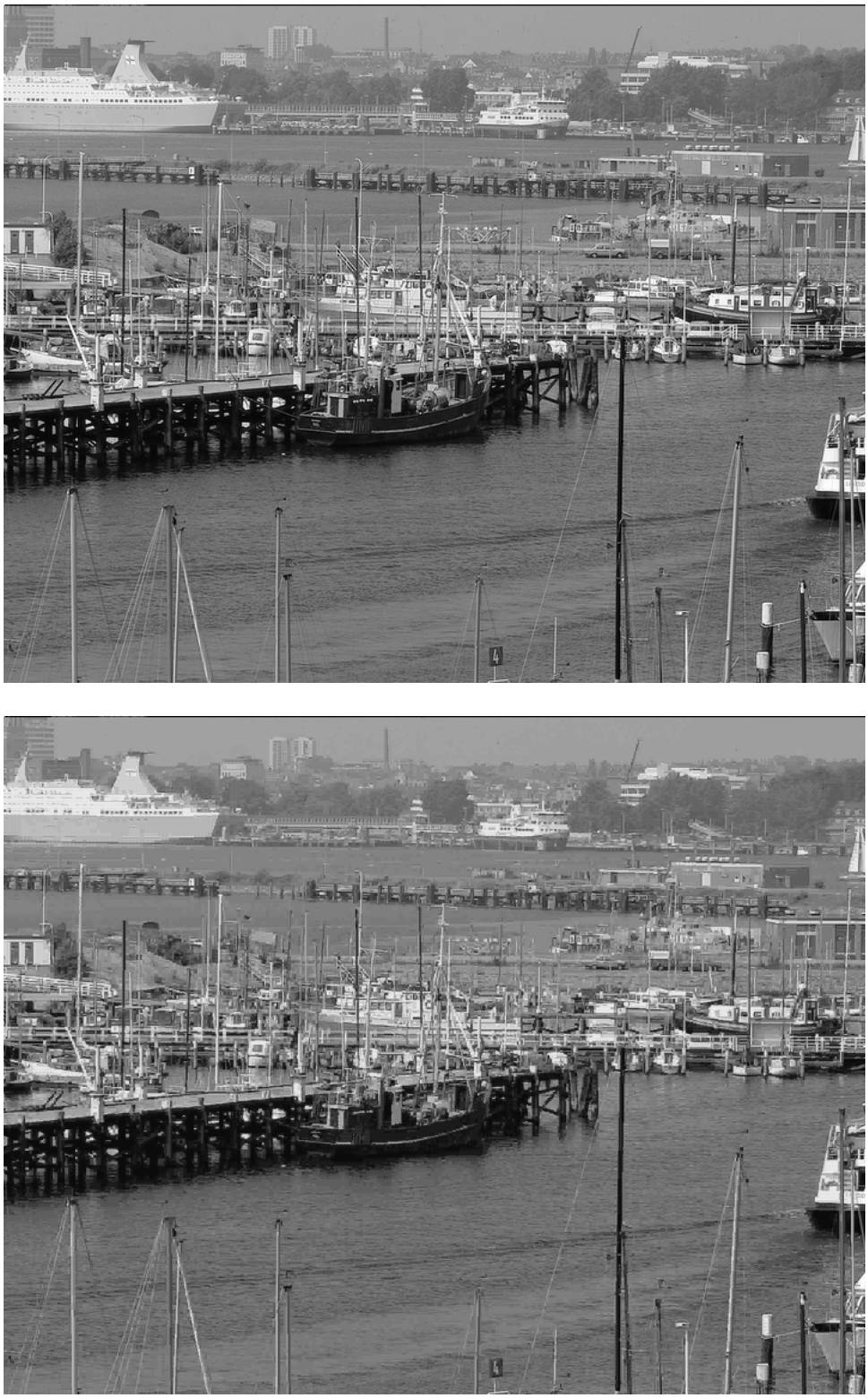


Figure 9: Image "Kiel" et sa reconstruction après compression

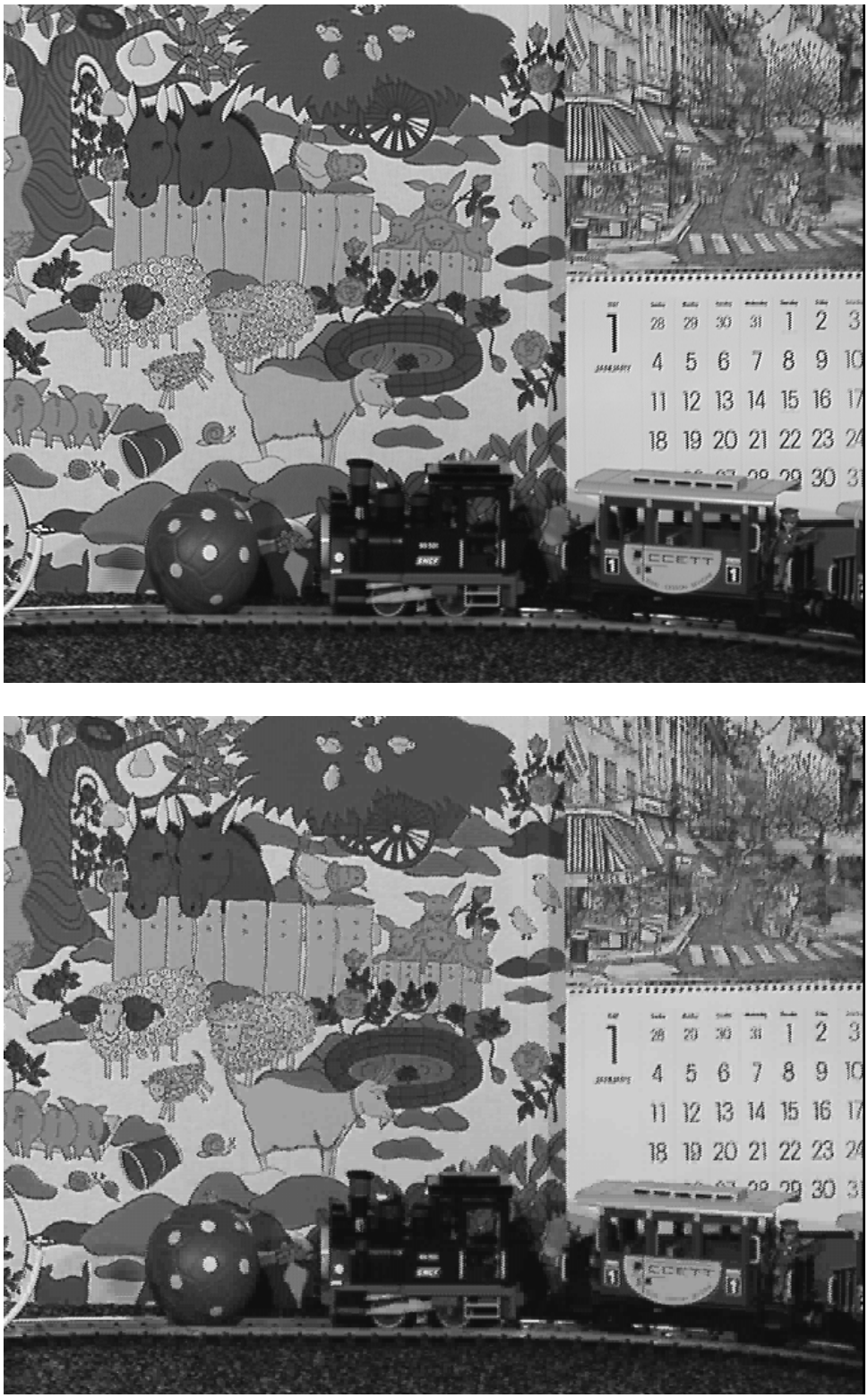


Figure 10: Image "mobile" et sa reconstruction après compression

Figure 11: Image "bateau" et sa reconstruction après compression

Chapitre 10:

SEPARATION DE SOURCES

Le problème de la séparation de sources est un problème fondamental en traitement du signal. Par exemple, dans le domaine biomédical, les mesures fournies par les capteurs sont généralement des mélanges de signaux issus de plusieurs sources indépendantes. Le problème consiste à extraire de ces mélanges les signaux utiles. On retrouve le même type de problème en traitement d'antenne pour la séparation de sources sonar ou radar, ou encore en traitement de la parole pour l'amélioration des signaux de diction (élimination des sources sonores parasites: autres locuteurs, bruits de moteurs dans un avion, ...).

Nous proposons dans ce chapitre un nouvel algorithme de séparation de sources. Cet algorithme exploite uniquement l'hypothèse d'indépendance des sources, et ne nécessite donc aucune hypothèse a priori sur les sources elles mêmes. Il est de plus capable de traiter des mélanges non-linéaires. Après un rappel de l'état de l'art dans ce domaine, nous exposerons l'algorithme proposé. Une étude expérimentale permettra ensuite de valider notre approche.

1 Introduction

La séparation de sources est un important problème de traitement du signal. Par exemple, dans le domaine bio-médical, les signaux fournis par les capteurs sont souvent des mélanges de plusieurs sources indépendantes. La séparation de sources consiste à extraire les signaux originaux de ces mélanges. Le même type de problème se retrouve en traitement d'antenne pour les signaux radar ou sonar par exemple, ou en traitement de la parole pour l'amélioration des signaux de diction (réduction des sources perturbatrices tels que les autres locuteurs, les bruits de moteur dans un cockpit, etc).

Comme l'a noté Jutten [*Jutten*91*a*], un système de séparation de sources existe probablement dans le système nerveux central, car les signaux qui circulent dans les nerfs sont généralement des mélanges d'informations hétérogènes. Par exemple, durant un mouvement, les fibres Ia et II transmettent au système nerveux central des mélanges d'informations relatives à la position et à la vitesse de l'articulation [*Roll*81].

La séparation de sources peut être réalisée de différentes façons selon la quantité et la nature des connaissances a priori dont on dispose. Ici, nous nous intéressons au cas où aucune information a priori sur les sources elles mêmes n'est disponible. Les seules hypothèses sont les suivantes:

- Nous connaissons une forme paramétrique du mélange.
- Les sources sont indépendantes (au sens mathématique du terme: séparabilité des densités de probabilité).
- Le nombre de capteurs est égal au nombre de sources.

Sous ces hypothèses, la séparation de sources est appelée séparation "aveugle", à cause de l'absence d'information sur les sources elles mêmes. Une solution au problème de séparation de sources sous des hypothèses aussi faibles n'a été trouvée que récemment par Jutten et Herault [*Jutten*88], pour les formes paramétriques linéaires. Dans le paragraphe suivant nous rappellerons brièvement leur méthode. Notre objectif dans ce chapitre est de proposer une nouvelle approche, basée sur la notion d'apprentissage par rétropropagation. Cette méthode diffère des précédentes par les points suivants:

- Elle est basée sur la minimisation d'une fonction de coût. En conséquence son fonctionnement est clair d'un point de vue théorique.
- Elle peut traiter des mélanges non-linéaires.

Dans la suite nous noterons n le nombre de capteurs (qui est égal au nombre de sources), $\vec{x}(t)$ les sources, $\vec{e}(t)$ les observations (signaux fournis par les capteurs), et $\vec{s}(t)$ les sorties du dispositif de séparation (l'index t est le temps: il sera supposé discret):

$$\vec{x}(t) = [x_1(t), ..., x_n(t)]^T$$
$$\vec{e}(t) = [e_1(t), ..., e_n(t)]^T$$
$$\vec{s}(t) = [s_1(t), ..., s_n(t)]^T$$

Le dispositif de séparation peut être vu comme une boîte noire, qui, à l'instant t, reçoit en entrée $\vec{e}(t)$ et fournit en sortie $\vec{s}(t)$. L'algorithme de séparation ajuste les paramètres internes du dispositif de séparation, de façon à ce que les composantes de $\vec{s}$ deviennent statistiquement indépendantes. Nous verrons plus loin que lorsque ceci est le cas, on retrouve en sortie du dispositif de séparation les sources originales, modulo une permutation et une distorsion.

Ouvrons ici une parenthèse pour signaler que nous n'avons pas adressé dans ce travail le problème de l'unicité de la solution. En termes plus clairs, on pourrait se poser la question suivante: est il possible que le dispositif de séparation fournisse des sorties indépendantes qui n'aient aucun rapport avec les sources originales? L'étude théorique à mener pour répondre à cette question nous semble particulièrement difficile, et peut être insoluble (sauf éventuellement dans le cas linéaire). Elle fera l'objet de travaux futurs. Signalons toutefois que ce problème n'a jamais été constaté dans nos expérimentations.

Il est intéressant de noter qu'un algorithme de séparation "aveugle" peut être vu comme un algorithme d'analyse en composantes indépendantes (INCA: INdependant Component Analysis). Ce concept d'analyse en composantes indépendantes a (à notre connaissance) été évoqué pour la première fois par Jutten. Le concept INCA est bien plus puissant que le concept traditionnel d'analyse en composantes principales (PCA: Principal Components Analysis), car INCA est une transformation qui réalise l'indépendance statistique (séparabilité des densités de probabilité), alors que PCA ne réalise que la décorrélation (séparabilité des moments croisés d'ordre 2).

Comme illustration, considérons le cas très simple d'un signal bi-dimensionnel à densité de probabilité uniforme à l'intérieur d'un parallélogramme (figure 1). PCA trouve une tranformation orthogonale (que l'on peut voir comme un changement de coordonnées) telle que la dispersion maximale est obtenue sur le premier axe. Dans ce nouveau système de coordonnées, les signaux S'_1 et S'_2 sont non-corrélés, mais sont toujours dépendants. Une façon simple d'être convaincu de cette dépendance est de

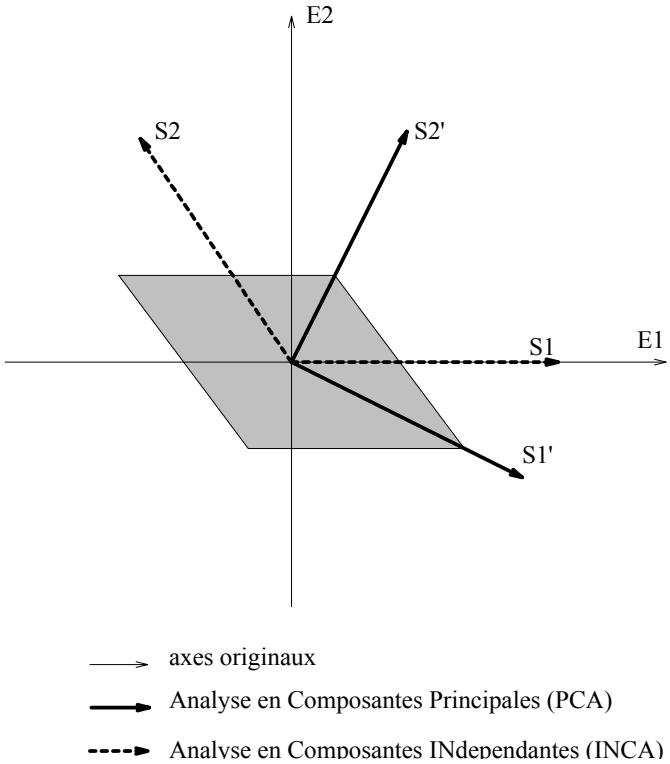


Figure 1: *Analyse en Composantes Principales (PCA) et Analyse en Composantes INdépendantes (INCA), dans un cas linéaire*

noter que la connaissance de S'_1 apporte une information sur S'_2, car les bornes de S'_2 dépendent de S'_1. Au contraire, INCA trouve un système de coordonnées dans lequel les signaux sont totalement indépendants. La connaissance de S_1 n'apporte aucune information sur S_2.

Considérons à présent le cas de la figure 2. Ici, aucune transformation linéaire ne peut réaliser l'indépendance. Une transformation non-linéaire, de façon à obtenir un nouveau système de coordonnées tel que celui qui est indiqué sur la figure, est requise. L'algorithme que nous proposons est capable de traiter des problèmes non-linéaires.

2 Travaux précédents

Le problème de la séparation "aveugle" a été résolu pour la première fois par Jutten et Hérault [*Jutten*88] [*Jutten*91*a*]. Leur algorithme est basé sur un réseau

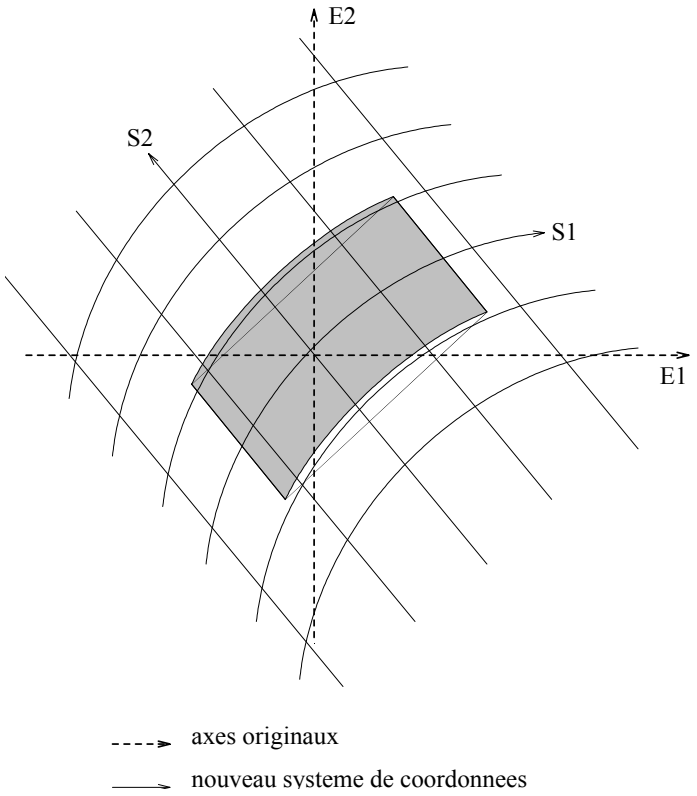


Figure 2: *Analyse en Composantes INdépendantes dans un cas non-linéaire*

de neurones linéaire avec des connexions récurrentes. Depuis, d'autres algorithmes qui seraient plus rapides ont été proposés (voir par exemple [*Comon*89]). La notion de rapidité doit toutefois être manipulée avec prudence, car l'algorithme de Jutten et Hérault présente l'avantage de ne nécessiter que des calculs localisés, et est donc particulièrement adapté à l'implantation sur machine parallèle.

Dans le cas linéaire, nous avons:

$$\vec{e}(t) = A\vec{x}(t)$$
$$\vec{s}(t) = F\vec{e}(t)$$

où A et F sont des matrices carrées nxn. L'algorithme de séparation ajuste la matrice F afin de rendre les composantes de $\vec{s}$ statistiquement indépendantes. Ces composantes sont alors égales aux sources originales, modulo une permutation et une dilatation. Cette indétermination est inévitable car on ne dispose d'aucune information a priori sur les sources. En effet, si les composantes de $\vec{x}$ sont indépendantes, alors les composantes de $DP\vec{x}$ le sont aussi (D étant une matrice diagonale, et P une matrice de permutation). Il est impossible de lever l'indétermination sans hypothèse suplémentaire.

- Dans [*Jutten*88], la matrice F est ajustée indirectement sous la forme $F = (I + C)^{-1}$ où I est la matrice identité, et C une matrice dont tous les termes diagonaux sont nuls. Pour un problème à 2 sources, l'itération s'écrit:

$$C(t) = C(t-1) + \mu \begin{pmatrix} 0 & f[s_1(t-1)]g[\breve{s}_2(t-1)] \\ f[s_2(t-1)]g[\breve{s}_1(t-1)] & 0 \end{pmatrix} \quad (1)$$

$$\vec{s}(t) = [I + C(t)]^{-1}\vec{e}(t)$$

  Les fonctions f et g sont des fonctions non-linéaires. Les auteurs suggèrent de prendre $f(y) = y^3$ et $g(y) = atan(10y)$. Dans l'équation (1), $\breve{s}_i(t)$ désigne $s_i(t)$ centré.

  Jutten et Hérault ont montré que la matrice C qui réalise la séparation est un état stable de l'algorithme. Cet algorithme ne minimise pas une fonction de coût, et son étude théorique n'est donc pas évidente. Toutefois, des recherches récentes, tant au niveau théorique qu'expérimental , ont montré que, dans certains cas (selon l'état initial et les statistiques des sources), l'algorithme peut ne pas converger vers la bonne solution (voir notamment [*Comon*91] et [*Sorouchyari*91]). Lorsque l'algorithme converge correctement (ce qui est le cas en général), on constate que cette convergence se décompose en 2 phases: une première phase, rapide, qui aboutit à la décorrélation, et une seconde phase, plus lente, qui réalise l'indépendance.

- La méthode de Comon [*Comon*89] est directe. Elle est basée sur l'idée d'établir un système d'équations entre les cumulants d'ordre 4 de $\vec{e}$ et les cumulants d'ordre 4 de $\vec{s}$. Ceci est possible à cause de la restriction à des formes paramétriques linéaires. Comon a montré que, sous l'hypothèse de linéarité, l'annulation des cumulants d'ordre 4 de $\vec{s}$ est suffisante pour assurer l'indépendance de ses composantes. L'algorithme résout le système d'équations par une méthode de moindre carrés (les inconnues sont les coefficients de la matrice F).

De premiers résultats d'études visant à l'extension de l'algorithme de Jutten aux mélanges convolutifs ont été publiés très récemment [*Jutten*91*b*]. A signaler également les travaux de Gaeta et Lacoume [*Gaeta*90] qui apportent un éclairage original à ces problèmes de séparation linéaire, et une première tentative de synthèse dans [*Duvaut*90].

3 Méthode proposée

3.1 Idée générale

Nous rappelons que nous observons une transformation instantannée $\vec{e} = v(\vec{x})$, éventuellement non-linéaire, de sources indépendantes $x_1, ..., x_n$. Nous connaissons une forme paramétrique de cette transformation. Notons w l'inverse de cette forme paramétrique (des exemples concrets seront donnés dans la partie "expérimentations"). Le problème est de calculer les paramètres de w de telle sorte que les composantes de $\vec{s} = w(\vec{e})$ soient indépendantes.

Notre objectif est de mettre au point un algorithme de séparation qui ne soit pas limité aux mélanges linéaires, et dont le fonctionnement soit clairement établi sur le plan théorique. Pour cela, on propose de procéder en deux étapes:

- Définition d'une mesure de la dépendance des composantes de $\vec{s}$.
- Mise au point d'une méthode de calcul des paramètres de w qui minimisent cette mesure.

La mesure de dépendance sera construite à partir des densités de probabilité. Pour la mise en œuvre pratique, nous en établirons une forme équivalente, qui s'exprime en fonction des moments. Pour minimiser cette mesure de dépendance, nous utiliserons un algorithme de gradient. Le dispositif de séparation est représenté figure 3 .

Nous nous intéresserons plus particulièrement au cas où la forme paramétrique inverse du mélange peut se mettre sous la forme d'un réseau de neurones multicouches. Le dimensionnement du réseau ainsi que le choix des fonctions d'activation des neurones seront alors conformes à la forme paramétrique inverse du mélange (des exemples concrets seront fournis dans la partie "expérimentations"). Cette forme paramétrique est en général connue car on connait les processus physiques qui interviennent dans le problème (par exemple les phénomènes d'atténuation en radiocommunications). Par contre, les valeurs numériques des paramètres sont inconnues. Signalons enfin que la connaissance de la forme paramétrique du mélange était également nécessaire dans les approches rappelées précédemment, mais cette forme ne pouvait être que linéaire.

Les réseaux de neurones multicouches sont des structures suffisamment générales pour représenter, ou tout au moins approximer, un grand nombre de formes paramétriques

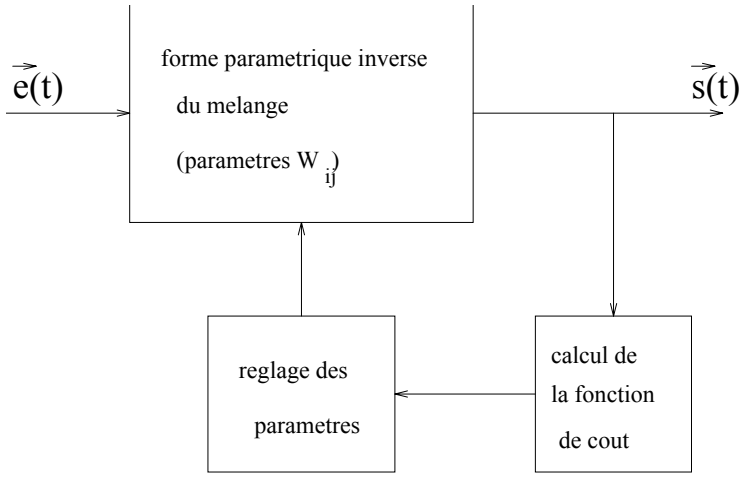


Figure 3: *Principe*

(les poids du réseau sont les paramètres). Pour le calcul du gradient, on pourra alors récupérer une partie de l'algorithme de rétropropagation. On est ici confronté à un problème très différent de ce qui se fait classiquement dans le domaine des réseaux multicouches, car l'erreur (mesure de dépendance) devra être définie par rapport aux sorties obtenues elles mêmes, et non pas par rapport à l'écart avec des sorties désirées. De plus, comme on manipule des phénomènes stochastiques, l'erreur ne pourra être qu'estimée, et l'estimateur devra avoir une structure telle qu'il permette le calcul des dérivées partielles intervenant dans l'algorithme de rétropropagation.

Signalons enfin que le cas des mélanges linéaires n'est pour l'algorithme proposé qu'un cas particulier (réseau à 2 couches et neurones totalement linéaires). D'autre part, lorsque la forme paramétrique inverse du mélange ne peut pas être approximée par un réseau multicouches, l'approche est toujours valable, mais il faut alors déterminer et programmer les équations de calcul du gradient pour cette forme spécifique.

3.2 Elaboration d'une mesure de dépendance

3.2.1 Notations

Pour la suite nous noterons:

$$\vec{\Theta}_i = [\underbrace{\overbrace{0,...,0}^{i-1},1,0,...,0}_{n}]^T$$

$$R_{\alpha_1...\alpha_n} = E\left\{S_1^{\alpha_1}...S_n^{\alpha_n}\right\}$$

$$\begin{aligned} M_{\alpha_1...\alpha_n} &= E\left\{S_1^{\alpha_1}...S_n^{\alpha_n}\right\} - E\left\{S_1^{\alpha_1}\right\}...E\left\{S_n^{\alpha_n}\right\} \\ &= R_{\alpha_1...\alpha_n} - R_{\alpha_1\vec{\Theta}_1}...R_{\alpha_n\vec{\Theta}_n} \end{aligned}$$

3.2.2 Définition d'une mesure de dépendance

Pour obtenir sur $\vec{s}$ une estimation de $\vec{x}$, il est nécessaire que les S_i soient indépendants. Une condition nécessaire et suffisante pour l'indépendance est:

$$\forall s_1,...,s_n \quad p_{S_1...S_n}(s_1,...,s_n) = p_{S_1}(s_1)...p_{S_n}(s_n) \tag{2}$$

Dans cette équation, p représente la densité de probabilité. Nous proposons d'utiliser l'expression suivante comme "mesure" du degré de dépendance:

$$\int_{-\infty}^{+\infty}...\int_{-\infty}^{+\infty}\left([p_{S_1...S_n}(s_1,...,s_n) - p_{S_1}(s_1)...p_{S_n}(s_n)] * \prod_{i=1}^{n}\frac{1}{\sqrt{2\pi}\sigma_i}e^{-\frac{S_i^2}{2\sigma_i^2}}\right)^2 ds_1...ds_n \tag{3}$$

On notera que nous avons introduit un filtrage gaussien sur le terme à intégrer (le symbole "$*$" désigne le produit de convolution). Ceci permet d'obtenir une mesure mieux adaptée au traitement de signaux réels (en particulier, on évite les variations brutales de la mesure de dépendance, et on atténue l'effet des erreurs d'arrondi ou de mesure). On évite également la divergence de l'intégrale lorsque les sorties sont discrètes.

Il semble laborieux d'exploiter la mesure de dépendance sous cette forme, car il serait nécessaire d'estimer les densités de probabilité et de réaliser une intégration.

Nous allons donc chercher une écriture différente de cette mesure en utilisant la transformation de Fourier (ce qui nous permettra de faire apparaître les moments). La transformée de Fourier d'une densité de probabilité est la fonction caractéristique:

$$\begin{aligned}\Phi_{S_1...S_n}(u_1,...,u_n) &= \int_{-\infty}^{+\infty}...\int_{-\infty}^{+\infty} p_{S_1...S_n}(s_1,...,s_n)e^{-j(u_1s_1+...+u_ns_n)}ds_1...ds_n \\ &= E\left\{e^{-j(S_1u_1+...+S_nu_n)}\right\}\end{aligned} \tag{4}$$

On pourra vérifier aisément que la transformée de Fourier de $p_{S_1}...p_{S_n}$ est $\Phi_{S_1}...\Phi_{S_n}$. En conséquence, d'après le théorême de Plancherel, et compte tenu des propriétés bien connues de la transformation de Fourier, la mesure (3) est égale à:

$$\int_{-\infty}^{+\infty}...\int_{-\infty}^{+\infty} \mid \Phi_{S_1...S_n}(u_1,...,u_n) - \Phi_{S_1}(s_1)...\Phi_{S_n}(s_n) \mid^2 e^{-\sigma_1^2u_1^2}...e^{-\sigma_n^2u_n^2}du_1...du_n \tag{5}$$

Une alternative au filtrage gaussien aurait été de supprimer l'influence des hautes fréquences sur la mesure, en limitant les bornes des intégrales.

3.2.3 Expression de la mesure de dépendance en fonction des moments

Nous allons à présent expliciter la mesure de dépendance en décomposant les fonctions caractéristiques en série de Mac Laurin autour de $\vec{0}$. Pour le problème qui nous intéresse, ce développement existe toujours, car nous traitons des processus réels, qui sont donc nécessairement bornés. Or, les moments de V.A. bornées existent, et l'existence des moments est suffisante pour que le développement en série de Mac Laurin soit défini ([*Roubine*80] p.137).

$$\Phi_{S_1...S_n}(u_1,...,u_n) = \sum_{\alpha_1...\alpha_n} \frac{1}{\alpha_1!...\alpha_n!}\frac{\partial^{\alpha_1+...+\alpha_n}\Phi_{S_1...S_n}}{\partial u_1^{\alpha_1}...\partial u_n^{\alpha_n}}(0,...,0)u_1^{\alpha_1}...u_n^{\alpha_n} \tag{6}$$

et

$$\Phi_{S_i}(u_i) = \sum_{\alpha_i=0}^{\infty}\frac{1}{\alpha_i!}\frac{\partial^{\alpha_i}\Phi_{S_i}}{\partial u_i^{\alpha_i}}(0)u_i^{\alpha_i} \tag{7}$$

Dans ces formules, par convention, la dérivée partielle d'ordre 0 d'une fonction est la fonction elle même.

En conséquence, on a:

$$\Phi_{S_1...S_n}(u_1,...,u_n) - \Phi_{S_1}(s_1)...\Phi_{S_n}(s_n) = \sum_{\alpha_1...\alpha_n} T_{\alpha_1...\alpha_n}u_1^{\alpha_1}...u_n^{\alpha_n} \tag{8}$$

avec:

$$T_{\alpha_1...\alpha_n} = \frac{1}{\alpha_1!...\alpha_n!}\left\{\frac{\partial^{\alpha_1+...+\alpha_n}\Phi_{S_1...S_n}}{\partial u_1^{\alpha_1}...\partial u_n^{\alpha_n}}(0,...,0) - \frac{\partial^{\alpha_1}\Phi_{S_1}}{\partial u_1^{\alpha_1}}(0)...\frac{\partial^{\alpha_n}\Phi_{S_n}}{\partial u_n^{\alpha_n}}(0)\right\} \quad (9)$$

L'équation (4) permet d'écrire:

$$\begin{aligned}\frac{\partial^{\alpha_1+...+\alpha_n}\Phi_{S_1...S_n}}{\partial u_1^{\alpha_1}...\partial u_n^{\alpha_n}}(0,...,0) &= (-j)^{\alpha_1+...+\alpha_n} E\left\{S_1^{\alpha_1}...S_n^{\alpha_n}\right\} \\ &= (-j)^{\alpha_1+...+\alpha_n} R_{\alpha_1...\alpha_n} \quad (10)\end{aligned}$$

et donc :

$$T_{\alpha_1...\alpha_n} = \frac{1}{\alpha_1!...\alpha_n!}(-j)^{\alpha_1+...+\alpha_n} M_{\alpha_1...\alpha_n} \quad (11)$$

La mesure du degré de dépendance s'écrit:

$$\int_{-\infty}^{+\infty}...\int_{-\infty}^{+\infty} \mid \sum_{\alpha_1...\alpha_n} T_{\alpha_1...\alpha_n} u_1^{\alpha_1}...u_n^{\alpha_n} \mid^2 e^{-\sigma_1^2 u_1^2}...e^{-\sigma_n^2 u_n^2} du_1...du_n \quad (12)$$

Soit encore :

$$\int_{-\infty}^{+\infty}...\int_{-\infty}^{+\infty}\left(\sum_{\alpha_1...\alpha_n}\sum_{\beta_1...\beta_n} T_{\alpha_1...\alpha_n} T^*_{\beta_1...\beta_n} u_1^{\alpha_1+\beta_1}...u_n^{\alpha_n+\beta_n}\right) e^{-\sigma_1^2 u_1^2}...e^{-\sigma_n^2 u_n^2} du_1...du_n \quad (13)$$

Et en utilisant la linéarité de l'intégration:

$$\sum_{\alpha_1...\alpha_n}\sum_{\beta_1...\beta_n} T_{\alpha_1...\alpha_n} T^*_{\beta_1...\beta_n} \int_{-\infty}^{+\infty}...\int_{-\infty}^{+\infty} u_1^{\alpha_1+\beta_1}...u_n^{\alpha_n+\beta_n} e^{-\sigma_1^2 u_1^2}...e^{-\sigma_n^2 u_n^2} du_1...du_n \quad (14)$$

Soit:

$$\sum_{\substack{\alpha_1,\beta_1 \\ \alpha_1+\beta_1\ pair}} ... \sum_{\substack{\alpha_n,\beta_n \\ \alpha_n+\beta_n\ pair}} T_{\alpha_1...\alpha_n} T^*_{\beta_1...\beta_n} J_{\alpha_1+\beta_1}(\sigma_1)...J_{\alpha_n+\beta_n}(\sigma_n) \quad (15)$$

Où $J_{2k}(\sigma)$ est calculé en annexe 1:

$$J_{2k}(\sigma) = \frac{(2k)!}{4^k k!}\frac{\sqrt{2\pi}}{\sigma^{2k-1}}$$

En utilisant (11), on obtient comme expression de la mesure de dépendance:

$$\boxed{\sum_{\alpha_1=0}^{\infty} ... \sum_{\alpha_n=0}^{\infty} \sum_{\beta_1=0}^{\infty} ... \sum_{\beta_n=0}^{\infty} G_{\alpha_1...\alpha_n,\beta_1...\beta_n} M_{\alpha_1...\alpha_n} M_{\beta_1...\beta_n}} \tag{16}$$

Où:

$$G_{\alpha_1...\alpha_n,\beta_1...\beta_n} = \prod_{i=1}^{n} g(\alpha_i, \beta_i, \sigma_i) \tag{17}$$

et:

$$\begin{aligned} g(\alpha,\beta,\sigma) &= \frac{1}{\alpha!\beta!}(-1)^{\frac{\alpha-\beta}{2}} J_{\alpha+\beta}(\sigma) && si\ \alpha+\beta\ est\ pair \\ &= 0 && sinon \end{aligned}$$

Lorsque l'un au moins des indices tend vers l'infini, le coefficient $G_{\alpha_1...\alpha_n,\beta_1...\beta_n}$ tend vers 0 (cf annexe 2). Ceci permet en pratique de limiter la sommation à un certain ordre.

Le tableau suivant contient $\frac{\sigma^{\alpha+\beta-1}}{\sqrt{2\pi}} g(\alpha,\beta,\sigma)$ pour les premières valeurs des indices ($m = 10^{-3}$, $\mu = 10^{-6}$):

	0	1	2	3	4	5	6	7	8
0	1	0	$-250m$	0	$31.2m$	0	$-2.60m$	0	163μ
1		$500m$	0	$-125m$	0	$15.6m$	0	$-1.30m$	0
2			$187m$	0	$-39.1m$	0	$4.56m$	0	-366μ
3				$52.1m$	0	$-9.11m$	0	977μ	0
4					$11.4m$	0	$-1.71m$	0	167μ
5						$2.05m$	0	-269μ	0
6							313μ	0	-36.4μ
7								41.6μ	0
8									4.87μ

3.2.4 Remarques

L'expression de la mesure de dépendance en fonction des moments, établie ci-dessus, est une forme quadratique définie positive (FQDF). Elle n'est donc nulle que lorsque

tous les $M_{\alpha_1 \ldots \alpha_n}$ sont nuls (pour la théorie des formes quadratiques, on pourra par exemple consulter [*Angot*82]). D'autre part, on rappelle que cette expression est strictement égale à (3), car aucune approximation n'a été faite durant la démonstration.

En pratique, le développement en série de Taylor doit être limité à un certain ordre K. Dans ce cas, en se référant à (12) et (16), la mesure de dépendance est:

$$\boxed{\sum_{\alpha_1+\ldots+\alpha_n \leq K} \sum_{\beta_1+\ldots+\beta_n \leq K} G_{\alpha_1 \ldots \alpha_n, \beta_1 \ldots \beta_n} M_{\alpha_1 \ldots \alpha_n} M_{\beta_1 \ldots \beta_n}} \tag{18}$$

Nous avons vérifié, jusquà l'ordre 10, que les valeurs propres de la forme quadratique obtenue sont bien positives (et l'on a donc bien une FQDF). Un résultat contraire aurait permi de détecter une erreur dans les calculs conduisant de (3) à (16).

3.3 Définition d'une fonction de coût

La séparation de sources va être réalisée par un réseau de neurones multicouches, dont nous notons $\vec{e}$ le vecteur d'entrée et $\vec{s}$ le vecteur de sortie. Pour l'apprentissage, nous utiliserons une méthode de gradient, ce qui nous permettra de récupérer une partie de l'algorithme de rétropropagation. Le problème se ramène donc à la définition d'une fonction de coût, et à l'expression des dérivées de cette fonction par rapport aux sorties du réseau. Comme l'algorithme de rétropropagation permet de calculer les dérivées des sorties par rapport aux poids, il sera possible d'en déduire les dérivées du coût par rapport aux poids.

Nous proposons d'utiliser la fonction de coût suivante:

$$C = \frac{1}{2} \left\{ \sum_{i=1}^{n} \mid E\{S_i\} \mid^2 + \sum_{i=1}^{n} \mid E\{{S_i}^2\} - 1 \mid^2 + \sum_{\alpha_1+\ldots+\alpha_n \leq K} \sum_{\beta_1+\ldots+\beta_n \leq K} G_{\alpha_1 \ldots \alpha_n, \beta_1 \ldots \beta_n} M_{\alpha_1 \ldots \alpha_n} M_{\beta_1 \ldots \beta_n} \right\} \tag{19}$$

Cette fonction de coût a été choisie de telle sorte que les contraintes suivantes tendent à être vérifiées:

(C1) $\forall i \in \{1, ..., n\}, \quad E\{S_i\} = 0$

(C2) $\forall i \in \{1, ..., n\}, \quad E\{S_i^2\} = 1$

Ces contraintes correspondent en quelque sorte à une normalisation des sorties du séparateur. Notons qu'une contrainte de type (C2) est absolument indispensable, car sinon la solution la plus simple pour fournir des sorties indépendantes serait de fournir des sorties identiquement nulles. De plus, la mesure de dépendance qui a été proposée n'est pas invariante en facteur d'échelle. Si les sorties tendent à avoir une variance unité, ceci n'est plus un problème.

Les valeurs des coefficients $G_{\alpha_1 ... \alpha_n, \beta_1 ... \beta_n}$ sont entièrement déterminées par les écarts types σ_i du filtrage gaussien. Comme les sorties S_i tendent à avoir la même variance (du fait de C1 et C2), les σ_i peuvent être choisis égaux. Nos expérimentations ont été menées avec $\sigma_i = 1$. Une solution qui pourrait améliorer la convergence, mais que nous n'avons pas évaluée expérimentalement, serait de commencer l'apprentissage avec une forte valeur des σ_i (par exemple $\sigma_i = 5$), et de réduire progressivement cette valeur en cours d'apprentissage. L'idée sous jacente est de faire débuter l'algorithme avec un filtrage grossier, et d'affiner progressivement la solution. Ceci pourrait également permettre d'éviter des minima locaux, car la fonction de coût varie en cours d'apprentissage.

Deux modes de fonctionnement sont envisageables: l'apprentissage global, et l'apprentissage continu.

L'apprentissage continu consiste à modifier les poids du réseau de neurones après chaque propagation d'un vecteur d'entrée $\vec{e}(t)$ à travers le réseau. Les moments sont alors estimés par filtrage passe-bas. Ce type d'apprentissage peut être intéressant dans le cas où les signaux arrivent en temps réel, et où les capacités mémoire sont limitées.

En mode global, un nombre fini d'échantillons a été préalablement enregistré. Les moments sont calculés par une moyenne sur tous ces échantillons, et les poids ne sont modifiés qu'après la propagation de tous les vecteurs $\vec{e}(t)$ à travers le réseau (afin de disposer du gradient exact).

Nous décrivons à présent l'algorithme d'apprentissage en mode global. Pour le mode continu, seul le résultat final sera donné, car, d'un point de vue formel, les calculs sont très similaires.

3.4 Apprentissage en mode global

En mode global, les moments peuvent être estimés par un moyennage sur les T échantillons disponibles:

$$\hat{R}_{\alpha_1 \ldots \alpha_n} = \frac{1}{T} \sum_{t=1}^{T} S_1^{\alpha_1}(t) \ldots S_n^{\alpha_n}(t)$$

et nous avons:

$$\hat{M}_{\alpha_1 \ldots \alpha_n} = \hat{R}_{\alpha_1 \ldots \alpha_n} - \hat{R}_{\alpha_1 \vec{\Theta}_1} \ldots \hat{R}_{\alpha_n \vec{\Theta}_n}$$

Il est donc possible d'écrire une estimation du coût:

$$\hat{C} = \frac{1}{2} \left\{ \sum_{i=1}^{n} \mid \hat{R}_{\vec{\Theta}_i} \mid^2 + \sum_{i=1}^{n} \mid \hat{R}_{2\vec{\Theta}_i} - 1 \mid^2 + \sum_{\alpha_1 + \ldots + \alpha_n \leq K} \sum_{\beta_1 + \ldots + \beta_n \leq K} G_{\alpha_1 \ldots \alpha_n, \beta_1 \ldots \beta_n} \hat{M}_{\alpha_1 \ldots \alpha_n} \hat{M}_{\beta_1 \ldots \beta_n} \right\} \tag{20}$$

Le problème que nous avons à résoudre est le calcul du gradient. On rappelle que les modifications des poids se font proportionellement au gradient (cf algorithme de rétropropagation). On peut écrire:

$$\frac{\partial \hat{C}}{\partial W_{ab}} = \sum_{t=1}^{T} \sum_{j=1}^{n} \frac{\partial \hat{C}}{\partial S_j(t)} \frac{\partial S_j(t)}{\partial W_{ab}}$$

L'algorithme de rétropropagation [*Rumelhart*86] indique comment calculer $\dfrac{\partial S_j(t)}{\partial W_{ab}}$.

Il nous reste donc à déterminer $\dfrac{\partial \hat{C}}{\partial S_j(t)}$.

$$\begin{aligned}
\frac{\partial \hat{C}}{\partial S_j(t)} &= \frac{\partial \hat{C}}{\partial \hat{R}_{\vec{\Theta}_j}} \frac{\partial \hat{R}_{\vec{\Theta}_j}}{\partial S_j(t)} + \frac{\partial \hat{C}}{\partial \hat{R}_{2\vec{\Theta}_j}} \frac{\partial \hat{R}_{2\vec{\Theta}_j}}{\partial S_j(t)} + \sum_{\alpha_1 + \ldots + \alpha_n \leq K} \frac{\partial \hat{C}}{\partial \hat{M}_{\alpha_1 \ldots \alpha_n}} \frac{\partial \hat{M}_{\alpha_1 \ldots \alpha_n}}{\partial S_j(t)} \\
&= \hat{R}_{\vec{\Theta}_j} \frac{\partial \hat{R}_{\vec{\Theta}_j}}{\partial S_j(t)} + \left(\hat{R}_{2\vec{\Theta}_j} - 1 \right) \frac{\partial \hat{R}_{2\vec{\Theta}_j}}{\partial S_j(t)} \\
&\quad + \sum_{\alpha_1 + \ldots + \alpha_n \leq K} \left(\sum_{\beta_1 + \ldots + \beta_n \leq K} G_{\alpha_1 \ldots \alpha_n, \beta_1 \ldots \beta_n} \hat{M}_{\beta_1 \ldots \beta_n} \right) \frac{\partial \hat{M}_{\alpha_1 \ldots \alpha_n}}{\partial S_j(t)}
\end{aligned} \tag{21}$$

Pour les dérivées par rapport à $S_j(t)$ on a:

$$\frac{\partial \hat{R}_{\vec{\Theta}_j}}{\partial S_j(t)} = \frac{1}{T}$$

$$\frac{\partial \hat{R}_{2\vec{\Theta}_j}}{\partial S_j(t)} = \frac{2}{T} S_j(t)$$

$$\begin{aligned}\frac{\partial \hat{M}_{\alpha_1 \ldots \alpha_n}}{\partial S_j(t)} &= \frac{\partial \hat{R}_{\alpha_1 \ldots \alpha_n}}{\partial S_j(t)} - \frac{\partial \hat{R}_{\alpha_j \vec{\Theta}_j}}{\partial S_j(t)} \prod_{\substack{l=1 \\ l \neq j}}^{n} \hat{R}_{\alpha_l \vec{\Theta}_l} \\ &= I\left(\alpha_j > 0\right) \left(\frac{1}{T} \alpha_j S_j^{\alpha_j - 1}(t) \left(\prod_{\substack{l=1 \\ l \neq j}}^{n} S_l^{\alpha_l}(t) - \prod_{\substack{l=1 \\ l \neq j}}^{n} \hat{R}_{\alpha_l \vec{\Theta}_l} \right) \right)\end{aligned}$$

où $I\left(\alpha_j > 0\right)$ vaut 1 si $\alpha_j > 0$, et 0 sinon.

Nous obtenons finalement:

$$\begin{aligned}&\frac{\partial \hat{C}}{\partial S_j(t)} = \\ &\quad \frac{1}{T} \left\{ \hat{R}_{\vec{\Theta}_j} + 2 \left(\hat{R}_{2\vec{\Theta}_j} - 1 \right) S_j(t) \right\} \\ &\quad + \frac{1}{T} \left\{ \sum_{\substack{\alpha_1 + \ldots + \alpha_n \leq K \\ \alpha_j > 0}} \alpha_j S_j^{\alpha_j - 1}(t) \left(\prod_{\substack{l=1 \\ l \neq j}}^{n} S_l^{\alpha_l}(t) - \prod_{\substack{l=1 \\ l \neq j}}^{n} \hat{R}_{\alpha_l \vec{\Theta}_l} \right) \left(\sum_{\beta_1 + \ldots + \beta_n \leq K} G_{\alpha_1 \ldots \alpha_n, \beta_1 \ldots \beta_n} \hat{M}_{\beta_1 \ldots \beta_n} \right) \right\}\end{aligned} \tag{22}$$

3.5 Apprentissage en mode continu

En mode continu, nous proposons d'estimer les moments à l'aide de filtres passe-bas:

$$\hat{R}_{\alpha_1 \ldots \alpha_n}(t) = (1 - \beta) S_1^{\alpha_1}(t) \ldots S_n^{\alpha_n}(t) + \beta \hat{R}_{\alpha_1 \ldots \alpha_n}(t-1)$$

Et nous avons toujours:

$$\hat{M}_{\alpha_1 \ldots \alpha_n}(t) = \hat{R}_{\alpha_1 \ldots \alpha_n}(t) - \hat{R}_{\alpha_1 \vec{\Theta}_1}(t) \ldots \hat{R}_{\alpha_n \vec{\Theta}_n}(t)$$

Il est possible d'écrire une estimation du coût:

$$\begin{aligned}\hat{C}(t) &= \frac{1}{2} \left\{ \sum_i \mid \hat{R}_{\vec{\Theta}_i}(t) \mid^2 + \sum_i \mid \hat{R}_{2\vec{\Theta}_i}(t) - 1 \mid^2 \right\} \\ &\quad + \frac{1}{2} \left\{ \sum_{\alpha_1 + \ldots + \alpha_n \leq K} \sum_{\beta_1 + \ldots + \beta_n \leq K} G_{\alpha_1 \ldots \alpha_n, \beta_1 \ldots \beta_n} \hat{M}_{\alpha_1 \ldots \alpha_n}(t) \hat{M}_{\beta_1 \ldots \beta_n}(t) \right\}\end{aligned} \tag{23}$$

Après un calcul formellement analogue au cas global, on obtient:

$$\frac{\partial \hat{C}(t)}{\partial S_j(t)} = \tag{24}$$
$$(1-\beta)\left\{\hat{R}_{\vec{\Theta}_j} + 2\left(\hat{R}_{2\vec{\Theta}_j} - 1\right) S_j(t)\right\}$$
$$+(1-\beta)\left\{\sum_{\substack{\alpha_1+\ldots+\alpha_n \leq K \\ \alpha_j > 0}} \alpha_j S_j^{\alpha_j - 1}(t) \left(\prod_{\substack{l=1 \\ l\neq j}}^{n} S_l^{\alpha_l}(t) - \prod_{\substack{l=1 \\ l\neq j}}^{n} \hat{R}_{\alpha_l \vec{\Theta}_l}\right) \left(\sum_{\beta_1+\ldots+\beta_n \leq K} G_{\alpha_1\ldots\alpha_n,\beta_1\ldots\beta_n} \hat{M}_{\beta_1\ldots\beta_n}\right)\right\}$$

4 Résultats expérimentaux

4.1 Conditions expérimentales

Nous présentons des expérimentations menées avec $n = 2$ sources en mode global. L'ordre du développement de Taylor est $K = 4$, et l'écart type du filtrage gaussien est $\sigma_1 = \sigma_2 = 1$

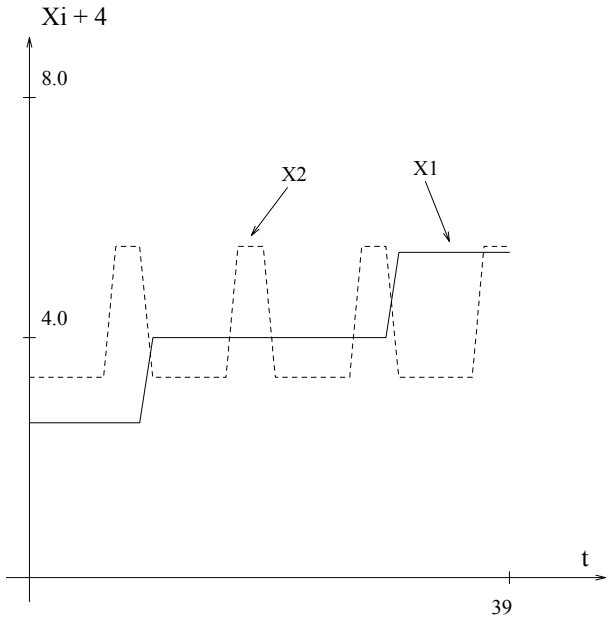


Figure 4: *Les sources*

Les résultats présentés ont été obtenus avec les sources utilisées par Comon pour ses expérimentations [*Comon*89]. Ces sources x_1 and x_2 ont les caractéristiques suivantes:

x_1	probabilité
$-\sqrt{2}$	0.25
0	0.5
$\sqrt{2}$	0.25

x_2	probabilité
$-\sqrt{\frac{0.3}{0.7}}$	0.7
$+\sqrt{\frac{0.7}{0.3}}$	0.3

Les sources (figure 4) sont centrées et de variance unité. Sur les figures, un offset de 4.0 a été appliqué, uniquement pour faciliter la visualisation.

4.2 Calcul automatique de la vitesse d'apprentissage

De premières expérimentations ont montré qu'il n'est pas possible d'obtenir une convergence avec une vitesse d'apprentissage constante.

Nous proposons donc une règle d'ajustement simple pour cette vitesse d'apprentissage, qui s'inspire de celle qui a été proposé au chapitre 3. Soulignons toutefois que nous étions alors en mode continu (ce qui nous a permi d'éviter une division par zéro lorsque le gradient devient faible), alors que les expérimentations présentées ici ont été réalisées en mode global. Nous allons donc devoir imposer une borne supérieure arbitraire sur la valeur de la vitesse d'apprentissage.

L'idée consiste à remplacer le paramètre "vitesse d'apprentissage" par un paramètre dont la valeur optimale dépend assez peu de l'application. Ce paramètre est la décroissance relative souhaitée sur le coût. Notons $\vec{g}$ le gradient, $\vec{W}$ le vecteur contenant les poids du réseau, α la vitesse d'apprentissage, C le coût, et d la décroissance relative souhaitée sur le coût. Une faible variation $\Delta\vec{W}$ des poids conduit à une variation $\Delta C = \vec{g}.\Delta\vec{W}$ du coût (par définition du gradient, et si la modification des poids est assez faible pour ne pas introduire d'effet non-linéaire notable). Mais, on rappelle que, selon l'algorithme de rétropropagation, la modification des poids est $\Delta\vec{W} = -\alpha\vec{g}$, où α est la vitesse d'apprentissage. Nous avons donc:

$$\Delta C = -\alpha\|\vec{g}\|^2$$

Si nous souhaitons obtenir $\Delta C = -d.C$, la vitesse d'apprentissage doit être:

$$\alpha = \frac{d.C}{\|\vec{g}\|^2}$$

Comme nous l'avons signalé plus haut, cette règle d'adaptation pose problème lorsque le gradient devient très faible. Nous avons choisi d'imposer une borne supérieure de 4.0 sur la valeur de α. Il est clair que la valeur de cette borne est arbitraire. Nous n'avons pas testé d'autres valeurs, celle-ci donnant de bons résultats en pratique. Notons toutefois qu'une meilleure stratégie serait sans doute de borner la norme de $\Delta\vec{W}$.

4.3 Les mélanges

- Mélange linéaire de Comon :
  C'est le mélange utilisé par Comon pour ces expérimentations [*Comon*89]:

$$\begin{pmatrix} e_1 \\ e_2 \end{pmatrix} = \begin{pmatrix} 0.32 & 0.95 \\ -0.95 & 0.32 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \tag{25}$$

  Une forme paramétrique inverse de ce mélange est:

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} e_1 \\ e_2 \end{pmatrix} \tag{26}$$

- Mélange non-linéaire en carré à 8 inconnues:

$$\begin{pmatrix} e_1 \\ e_2 \end{pmatrix} = \begin{pmatrix} 0.32 & 0.95 \\ -0.95 & 0.32 \end{pmatrix} \begin{bmatrix} sgn(.)[.]^2 \\ sgn(.)[.]^2 \end{bmatrix} \begin{pmatrix} 0.32 & 0.95 \\ -0.95 & 0.32 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \tag{27}$$

  Ce mélange consiste donc à mélanger les sources avec une transformation linéaire, à élever les composantes du vecteur résultat au carré en préservant le signe, et à mélanger à nouveau avec une tranformation linéaire. Une forme paramétrique inverse de ce mélange est:

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{bmatrix} sgn(.)\sqrt{|\,.\,|} \\ sgn(.)\sqrt{|\,.\,|} \end{bmatrix} \begin{pmatrix} e & f \\ g & h \end{pmatrix} \begin{pmatrix} e_1 \\ e_2 \end{pmatrix} \tag{28}$$

Le perceptron multicouches utilisé pour séparer le mélange non-linéaire est représenté figure 5. Les deux neurones de la couche intermédiaire sont des sommateurs pondérés suivis d'une fonction de transition en $sgn(.)\sqrt{|.|}$. Les deux neurones de la couche de sortie sont des sommateurs pondérés. Les 8 paramètres de la forme paramétrique

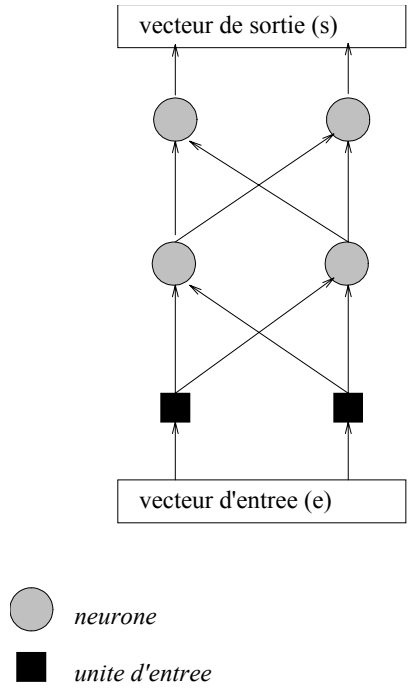


Figure 5: *Perceptron multicouches pour la séparation du mélange non-linéaire. Les 8 inconnues de la forme paramétrique inverse correspondent aux 8 poids du réseau*

inverse du mélange sont les 8 poids du réseau.

On trouvera également en annexe 3 les résultats du traitement d'un mélange non-linéaire en racine carrée à 8 inconnues.

4.4 Séparation du mélange linéaire

Le réseau de neurones est à 2 couches, avec 2 entrées et 2 sorties. Le modèle du neurone est linéaire (simple sommateur pondéré). La décroissance relative demandée sur l'erreur est $d = 10\%$.

La séparation est très rapide et ne nécessite que de 50 à 100 itérations. De nombreuses expérimentations ont été réalisées avec diverses initialisations, et aucun problème de minimum local n'a été détecté.

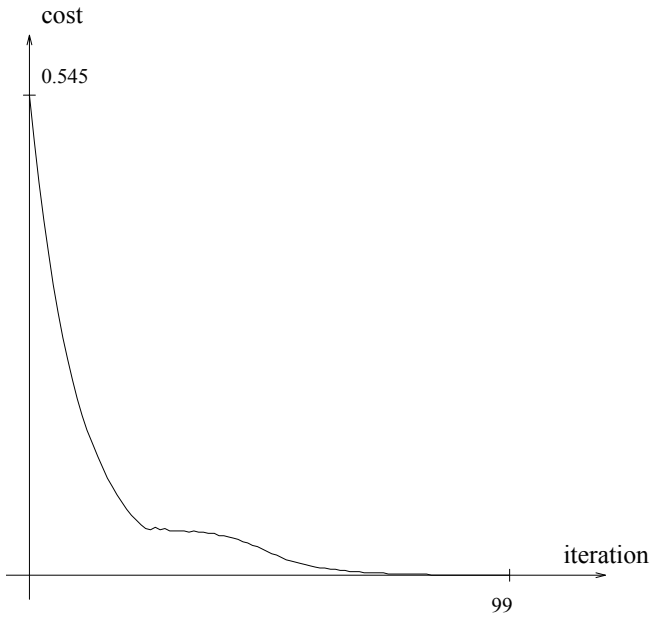


Figure 6: *Evolution du coût pendant l'apprentissage (mélange linéaire)*

Un exemple de courbe d'apprentissage est fourni figure 6. On peut distinguer trois zones: une première zone de décroissance rapide du coût (jusqu'à l'itération 20), puis une zone d'apprentissage difficile (itération 20 à 40), et finalement une zone de décroissance rapide du coût jusqu'à la séparation parfaite. Nous reviendrons sur ce point, car le même phénomène a été constaté dans le cas non-linéaire.

4.5 Séparation d'un mélange non-linéaire en carré à 8 inconnues

Un exemple d'évolution du coût pendant l'apprentissage est indiqué sur la figure 7. La décroissance relative demandée sur le coût est $d = 4\%$.

Comme dans le cas linéaire, la courbe d'apprentissage peut être divisée en 3 phases. Durant la première phase (les 100 premières itérations), une décroissance rapide de l'erreur est observée. Puis, on observe une phase d'apprentissage difficile (itérations 100 à 400), où le coût décroît très lentement, et parfois oscille légèrement. Durant cette phase, la vitesse d'apprentissage sature souvent sur sa borne maximale. A l'intérieur de l'espace paramétrique (les paramètres étant les poids du réseau) on se trouve donc dans une zone à très faible gradient. Cette zone contient probablement une forte densité de minima locaux. Toutefois, ces minima ne semblent pas trop

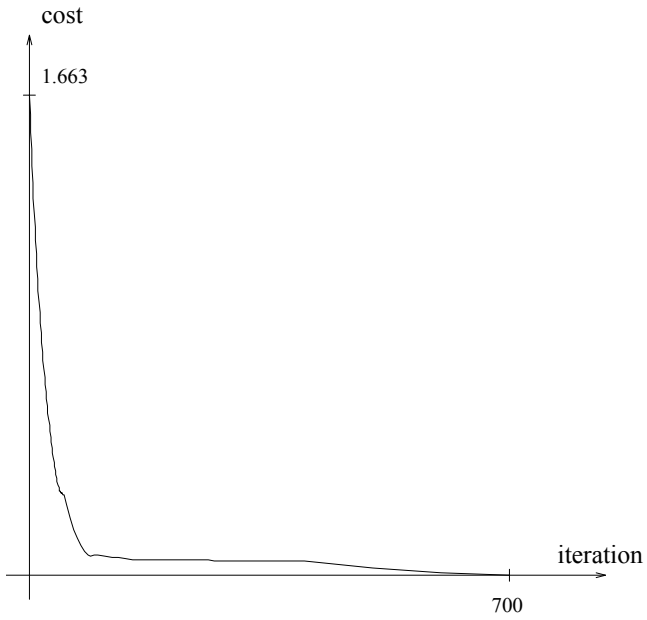


Figure 7: *Evolution du coût pendant l'apprentissage (mélange non-linéaire)*

profonds car l'algorithme n'a pas été définitivement piégé dans l'un d'eux. Finalement, une troisième phase de décroissance rapide du coût est observée (itérations 400 à 700).

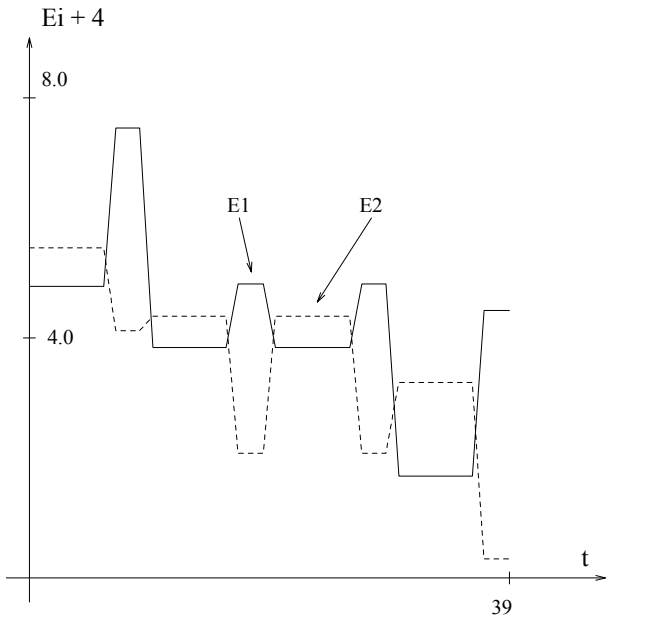


Figure 8: *signaux d'entrée du réseau de neurones (mélange des sources)*

La figure 8 montre le mélange des sources (signaux d'entrée du réseau). Les signaux de sortie en début et en fin d'apprentissage sont indiqués sur les figures 9 et 10. Cette dernière figure confirme que la séparation a réussi, car l'on retrouve en sortie

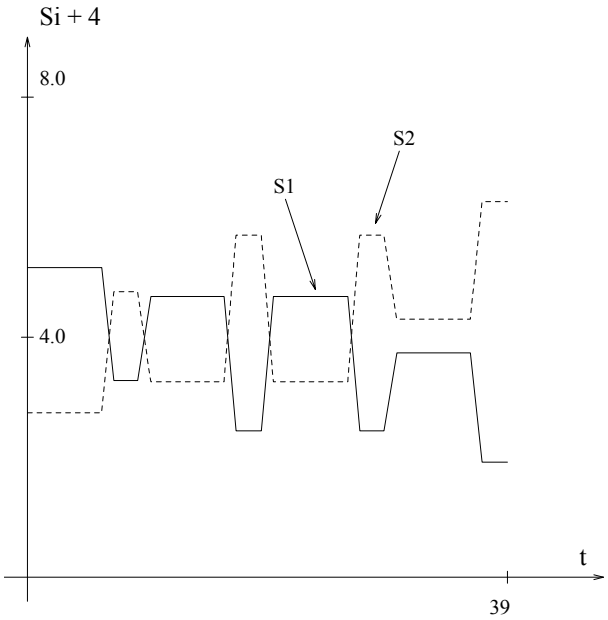


Figure 9: *signaux de sortie du réseau de neurones au début de l'apprentissage*

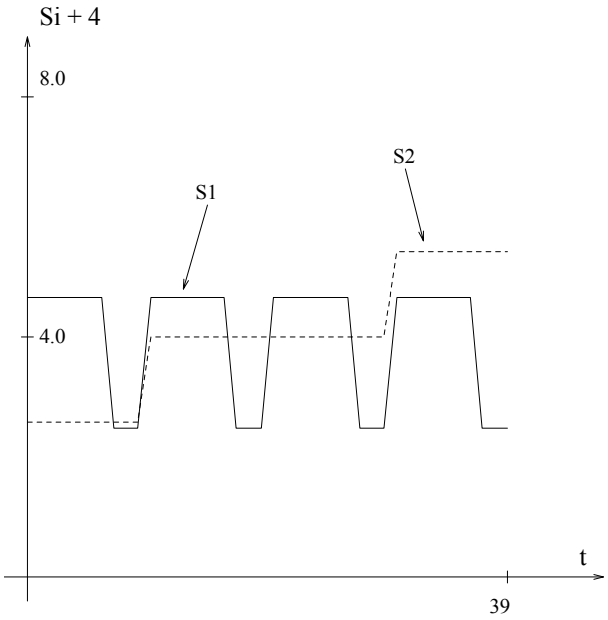


Figure 10: *signaux de sortie du réseau de neurones à la fin de l'apprentissage*

du réseau de neurones les signaux originaux, à une permutation et une négation près:

$$\begin{aligned} S_1(t) &= -X_2(t) \\ S_2(t) &= X_1(t) \end{aligned}$$

Pour donner une idée du temps de calcul requis, la séparation de ce mélange

non-linéaire a demandé 90 secondes de calcul sur une station de travail de type Sun4 (avec un programme écrit en C, mais pas du tout optimisé).

4.6 Séparation de mélange non-linéaire bruité

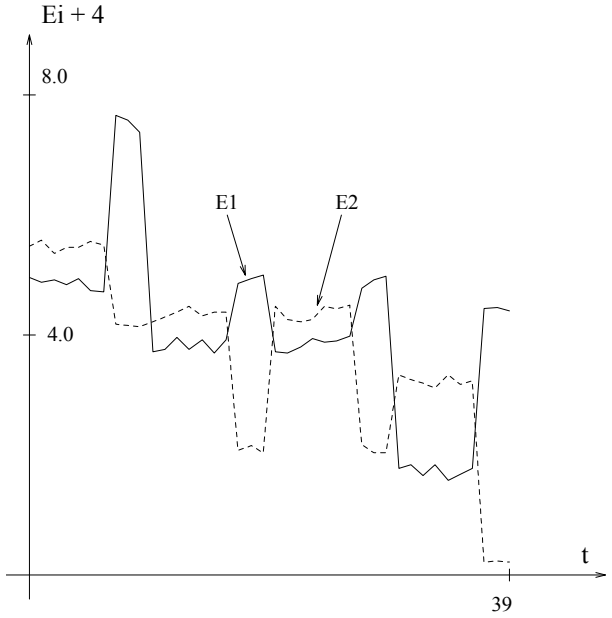


Figure 11: *signaux d'entrée du réseau de neurones, dégradés par un bruit*

Examinons à présent la robustesse de l'algorithme par rapport au bruitage des observations. Le mélange est le même que dans le paragraphe précédent, mais les signaux d'entrée du réseau sont dégradés par un bruit blanc additif à densité de probabilité uniforme dans $[-0.15, +0.15]$. La décroissance relative demandée sur le coût est toujours $d = 4\%$. Les observations (signaux d'entrée du réseau) sont indiquées sur la figure 11 et la courbe d'apprentissage obtenue sur la figure 12. On observe toujours une segmentation de cette courbe en trois phases. Cette courbe montre le succès de la séparation, malgré la présence simultannée de non-linéarités et de bruit.

La figure 13 montre les signaux de sortie du réseau en fin d'apprentissage. Ces signaux correspondent bien aux sources originales, ce qui confirme le succès de la séparation. Bien entendu, ces signaux sont dégradés par du bruit, car notre algorithme est un algorithme de séparation, et non pas un algorithme de réduction de bruit. Il est d'ailleurs clair que la réduction de bruit est absolument impossible sans hypothèse a priori sur le bruit ou sur les sources. Un grand nombre de méthodes de réduction de bruit ont été proposées dans la littérature. Si le bruit est important, on

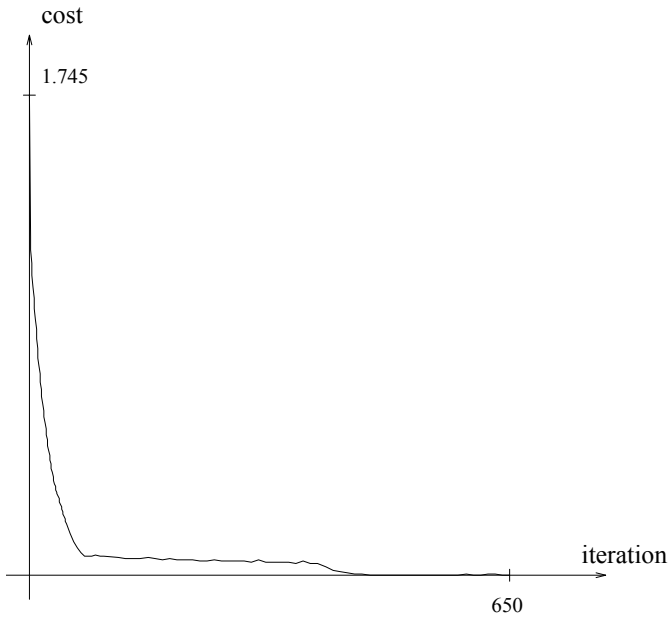


Figure 12: *Courbe d'apprentissage en présence de bruit*

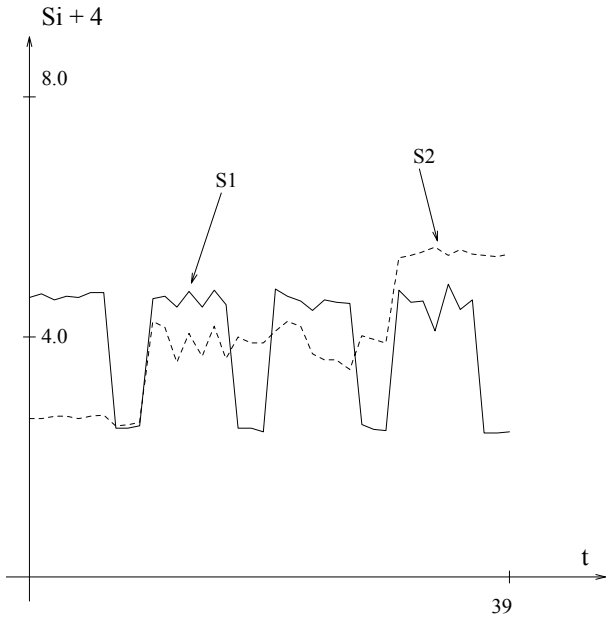


Figure 13: *Signaux de sortie du réseau de neurones en fin d'apprentissage (dans le cas non-linéaire bruité)*

peut envisager de placer un tel dispositif en amont ou en aval du séparateur (selon la nature des hypothèses a priori disponibles ou assumées).

5 Conclusion

Nous avons proposé un algorithme de séparation "aveugle ", capable de traiter des mélanges non-linéaires. Des résultats expérimentaux ont été fournis pour valider et illustrer la méthode proposée. D'autres expérimentations, avec d'autres sources et d'autres mélanges ont été réalisées et confirment les résultats énoncés. Le blocage définitif dans un minimum local semble assez rare, mais la phase de l'apprentissage correspondant à une décroissance très lente du coût peut quelquefois être assez longue.

De bons résultats sont obtenus, même dans les cas non-linéaires, ainsi que nous l'avons montré dans la partie "expérimentations". De plus, la robustesse de l'algorithme face au bruitage des observations a été montrée. En présence de bruit, l'algorithme arrive toujours à séparer les sources. Mais il ne réduit évidemment pas le bruit, car aucune hypothèse sur le bruit ni sur les sources n'a été assumée.

La principale limite de l'algorithme proposé, dans son état actuel, est le fait que la convergence n'est pas garantie, car les minima locaux sont toujours possibles. Il faut toutefois noter que nous avons expérimenté une situation difficile car les poids initiaux du réseau de neurones étaient choisis totalement aléatoires. En pratique, on a parfois une idée a priori de l'ordre de grandeur, et par exemple du signe des paramètres, ce qui permet de partir d'un état plus proche de la solution. De plus, une fois que l'algorithme a trouvé la solution, il n'a plus qu'à suivre la lente évolution temporelle de ces paramètres (par exemple variations lentes des caractéristiques d'un canal de transmission). On peut alors supposer que l'on se trouve en permanence à proximité de la solution. D'autre part, l'idée que nous avons émise de réduction progressive de la puissance du filtrage gaussien reste à évaluer expérimentalement. Cela pourrait être un atout pour l'évitement des minima locaux car la fonction de coût serait modifiée en permanence durant l'apprentissage.

D'autre part, diverses extensions de l'algorithme sont à étudier, notamment l'extension aux mélanges convolutifs (car dans beaucoup d'application réelles il est difficile de négliger la nature convolutive des canaux de transmission), et l'extension aux signaux complexes (pour le traitement de signaux modulés en phase par exemples).

Revenons enfin sur le phénomène de segmentation de la courbe d'apprentissage en trois phases, qui a été noté dans toutes les configurations expérimentales. Une recherche théorique de la cause de ce phénomène apporterait peut-être une meilleure compréhension de l'algorithme et pourrait susciter des améliorations.

Pour terminer, nous souhaitons mettre en valeur une différence fondamentale entre l'algorithme proposé et les mises en œuvre traditionnelles de la rétropropagation.

En effet, la fonction de coût à minimiser n'est nullement une mesure de l'écart entre les sorties obtenues et les sorties correctes. Dans un problème de séparation "aveugle", les sorties correctes sont totalement inconnues. Notre fonction de coût est entièrement bâtie sur les sorties obtenues elles mêmes. Dans ce contexte, la rétropropagation est un algorithme d'apprentissage non-supervisé.

Annexe 1 : Moments d'une gaussienne (rappels)

Le problème est de déterminer la valeur suivante:

$$I_{2k}(\sigma) = \frac{1}{\sqrt{2\pi}\sigma} \int_{-\infty}^{+\infty} x^{2k} e^{-\frac{x^2}{2\sigma^2}} dx$$

On a :

$$I_{2k}(\sigma) = \frac{-\sigma^2}{\sqrt{2\pi}\sigma} \int_{-\infty}^{+\infty} x^{2n-1} \left(-\frac{x}{\sigma^2}\right) e^{-\frac{x^2}{2\sigma^2}} dx$$

Et en intégrant par parties, on obtient :

$$I_{2n}(\sigma) = \frac{-\sigma^2}{\sqrt{2\pi}\sigma} \left\{ \left[x^{2n-1} e^{-\frac{x^2}{2\sigma^2}} \right]_{-\infty}^{+\infty} - \int_{-\infty}^{+\infty} (2n-1) x^{2n-2} e^{-\frac{x^2}{2\sigma^2}} dx \right\}$$

Soit :

$$I_{2k} = \sigma^2 (2k-1) I_{2k-2}$$

Comme $I_0(\sigma) = 1$, on en déduit :

$$\begin{aligned} I_{2k} &= \sigma^{2k}(2n-1)(2n-3)...1 \\ &= \sigma^{2k} \frac{(2k)!}{2^k k!} \end{aligned}$$

On vérifiera aisément que $I_{2k+1}(\sigma)$ est nul, car on intègre alors une fonction impaire.

Pour le problème qui nous intéresse, nous avons à calculer :

$$\begin{aligned} J_{2k}(\sigma) &= \int_{-\infty}^{+\infty} x^{2k} e^{-\sigma^2 x^2} dx \\ &= \sqrt{2\pi}\sigma I_{2k}\left(\frac{1}{\sqrt{2}\sigma}\right) \end{aligned}$$

Et on a donc :

$$\boxed{J_{2k}(\sigma) = \frac{(2k)!}{4^k k!} \frac{\sqrt{2\pi}}{\sigma^{2k-1}}}$$

Annexe 2 : Comportement des coefficients de la mesure de dépendance

On rappelle que l'on a l'approximation suivante pour la factorielle :

$$\alpha! \simeq \sqrt{2\pi\alpha}\ \alpha^{\alpha} e^{-\alpha} \left(1 + \frac{1}{12\alpha}\right)$$

Cette approximation est valable, même pour les faibles valeurs de α (pour $\alpha = 1$, l'erreur relative n'est que de 10^{-3}).

L'expression de $G_{\alpha_1 \ldots \alpha_n, \beta_1 \ldots \beta_n}$, est un produit de termes du type :

$$\frac{1}{\alpha!\beta!}(-1)^{\frac{\alpha-\beta}{2}} \frac{(\alpha+\beta)!}{2^{\alpha+\beta}\left(\frac{\alpha+\beta}{2}\right)!} \frac{\sqrt{2\pi}}{\sigma^{\alpha+\beta-1}}$$

En remplaçant les factorielles par leurs approximations, on obtient:

$$(-1)^{\frac{\alpha-\beta}{2}} \frac{\pi\sqrt{2\pi}}{\sigma^{\alpha+\beta-1}} \frac{(\alpha+\beta)^{\frac{\alpha+\beta}{2}}}{\alpha^{\alpha-1}\beta^{\beta-1}} \left(\frac{e}{2}\right)^{\frac{\alpha+\beta}{2}} \frac{1 + \frac{1}{12(\alpha+\beta)}}{\left(1+\frac{1}{12\alpha}\right)\left(1+\frac{1}{12\beta}\right)\left(1+\frac{1}{6(\alpha+\beta)}\right)}$$

Lorsque α ou β tend vers l'infini, cette expression se comporte comme:

$$\frac{(\alpha+\beta)^{\frac{\alpha+\beta}{2}}}{\alpha^{\alpha-1}\beta^{\beta-1}}$$

Par exemple, pour β fixé et α tendant vers l'infini, cette fraction se comporte comme: $\frac{\alpha^{\frac{\alpha}{2}}}{\alpha^{\alpha}}$, c'est à dire comme $\alpha^{-\frac{\alpha}{2}}$. Les coefficients de la mesure de dépendance tendent donc bien vers 0 pour les fortes valeurs des indices (plus précisément, il suffit que l'un des indices devienne important pour que le coefficient tende vers 0).

Annexe 3: Traitement d'un mélange non-linéaire bruité en racine carrée à 8 inconnues

Nous présentons les résultats du traitement du mélange suivant:

$$\begin{pmatrix} e_1 \\ e_2 \end{pmatrix} = \begin{pmatrix} 0.32 & 0.95 \\ -0.95 & 0.32 \end{pmatrix} \begin{bmatrix} sgn(.)\sqrt{|.|} \\ sgn(.)\sqrt{|.|} \end{bmatrix} \begin{pmatrix} 0.32 & 0.95 \\ -0.95 & 0.32 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \quad (29)$$

Ce mélange consiste donc à mélanger les sources par une transformation linéaire, à prendre la racine carrée de la valeur absolue des composantes du vecteur résultant (en préservant le signe), et à mélanger à nouveau avec une transformation linéaire. Les sources sont toujours les sources de Comon.

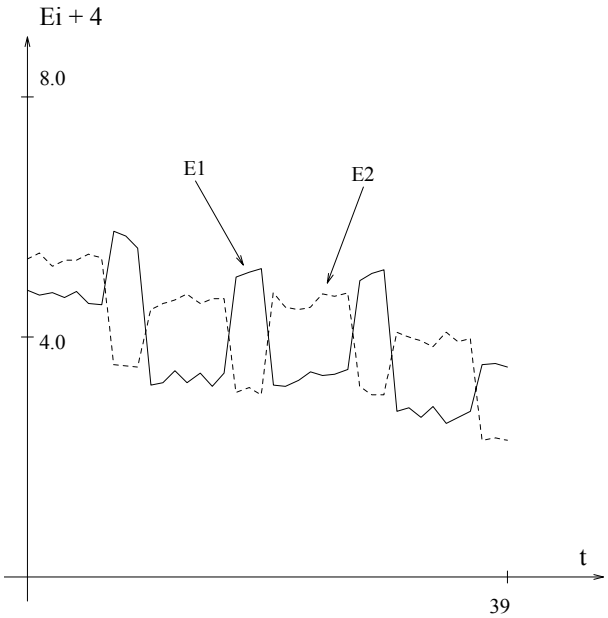


Figure 14: *Mélange en racine carrée bruité (observations)*

Les composantes du vecteur $\vec{e}$ ont de plus été bruitées par un bruit blanc uniforme dans $[-0.15, +0.15]$, ce qui donne les signaux de la figure 14. Le réseau de neurones est le même que celui de la figure 5, à la différence près que les neurones de la couche intermédiaire ont maintenant une fonction de transition en $sgn(.)[.]^2$. La décroissance relative demandée sur le coût est toujours $d = 4\%$. La figure 15 indique l'évolution du coût en cours d'apprentissage, et la figure 16 indique les signaux de sortie du réseau de neurones en fin d'apprentissage. On peut donc tirer les mêmes conclusions que pour le mélange en carré.

Ce succès de la séparation pour un mélange en racine carrée est intéressant

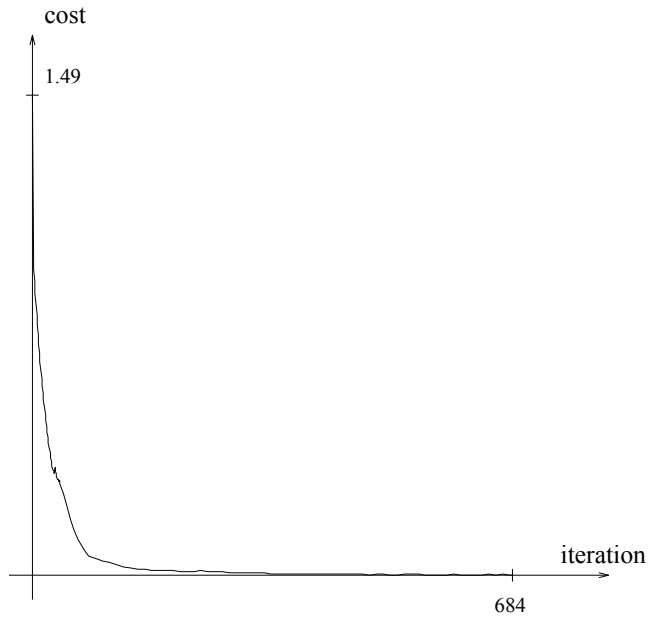


Figure 15: *Courbe d'apprentissage sur le mélange en racine carrée bruité*

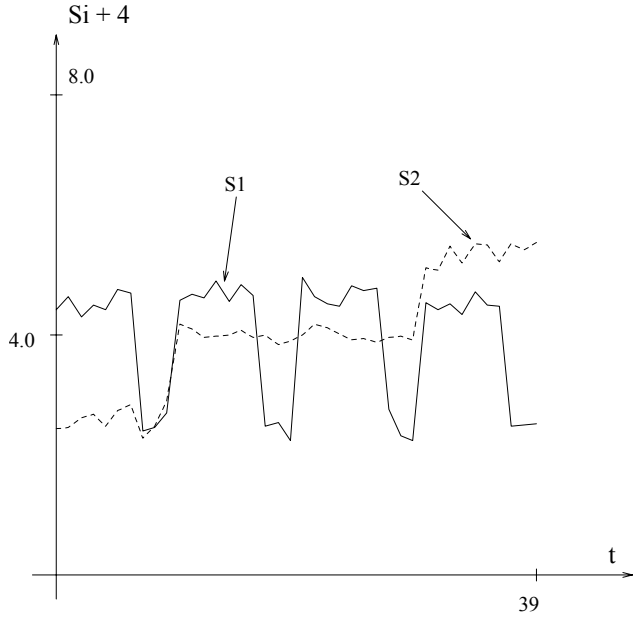


Figure 16: *Signaux de sortie du réseau en fin d'apprentissage (cas du mélange en racine carrée bruité)*

car il permet de répondre à une objection que l'on pourrait soulever concernant l'intérêt de la démonstration conduisant de la formule (3) à la formule (16). En effet, on pourrait envisager une approche plus simple consistant à prendre comme mesure de dépendance la somme des carrés des moments $M_{\alpha_1 \ldots \alpha_n}$. Une première version de notre méthode qui prenait comme mesure de dépendance cette expression simplifiée

s'était montrée tout à fait incapable de réaliser la séparation d'un mélange en racine carrée, même dans un cas non bruité. Ceci est dû à l'absence d'amortissement des moments d'ordre élevé dans cette mesure de dépendance simplifiée, qui, combinée aux élévations au carré dans le réseau de neurones, produit des variations brutales du gradient (d'où un apprentissage particulièrement instable). De plus, l'égalité entre (3) et (16) nous permet de disposer d'un puissant paramètre de contrôle (l'écart type du filtrage gaussien), que nous envisageons d'exploiter au mieux dans une version future.

Références

[*Angot*82] André ANGOT
"Compléments de mathématiques"
MASSON 1982 - 6^e édition

[*Comon*89] P. COMON
"Séparation de mélanges de signaux"
Douzième colloque GRETSI, Juan Les Pins, 12 au 16 juin 1989

[*Comon*91] P. COMON, C. JUTTEN, J. HERAULT
"Blind separation of sources, Part II: Problem statement"
Signal processing, vol 24, n^o1, July 1991

[*Duvaut*90] P. DUVAUT
"Principe des méthodes de séparation de sources
fondées sur les moments d'ordre supérieur"
Traitement du signal, vol 7, n^o5, 1990

[*Gaeta*90] M. GAETA, J.L. LACOUME
"Estimateurs du maximum de vraissemblance étendus à la séparation
de sources non gaussiennes"
Traitement du signal, vol 7, n^o5, 1990

[*Jutten*88] Christian JUTTEN, Jeanny HERAULT
"Une solution neuromimétique au problème de séparation de sources"
Traitement du Signal, vol. 5, n^o6, 1988

[*Jutten*91*a*] C. JUTTEN, J. HERAULT
"Blind separation of sources,
Part I: An adaptative algorithm based on a neuromimetic architecture"
Signal processing, vol 24, n^o1, July 1991

[*Jutten*91*b*] C. JUTTEN, L. NGUYEN THI, J. CAELEN, E. DIJKSTRA, E. VITTOZ
"Blind separation of sources:
An algorithm for separation of convolutive mixtures"
International Signal Processing Workshop on Higher Order Statistics
Chamrousse, France, July, 10-12th, 1991, pp273-276

[*Roll*81] J.P. ROLL
"Contribution de la proprioception musculaire à la perception
et au contrôle du mouvement chez l'homme"
Thèse de doctorat d'état, Univ. d'Aix-Marseille I, 1981

[*Roubine*80] E. ROUBINE, "Probabilités"
Ecole Supérieure d'Electricité - 1980 - Cours n^o 2798

[*Rumelhart*86] D.E. RUMELHART, G.E. HINTON, R.J. WILLIAMS
"Learning internal representations by error backpropagation"
Parallel Distributed Processing, D.E. RUMELHART and J.L. Mc CLELLAND
Chap8, Bradford book - MIT Press - 1986

[*Sorouchyari*91] E. SOROUCHYARI
"Blind separation of sources, Part III: Stability analysis"
Signal processing, vol 24, n^o1, July 1991

CONCLUSION

Les travaux relatés dans ce mémoire s'étendent suivant deux axes:

- Un axe théorique, selon lequel nous avons proposé des améliorations des algorithmes existants (réglage automatique des paramètres d'apprentissage), et mis au point de nouveaux algorithmes d'apprentissage.
- Un axe applicatif, selon lequel nous avons traité, par une approche neuronale, différents problèmes de traitement de l'image et du signal (de la segmentation à la séparation de sources, en passant par la reconnaissance de formes et la compression de données).

Cette diversité des problèmes traités avait pour but de montrer dans quelle mesure les réseaux de neurones peuvent être intéressants du point de vue industriel (aussi bien pour la résolution d'applications que pour la mise au point de nouveaux algorithmes). Les travaux relatés permettent de tirer les conclusions suivantes:

1. Les réseaux de neurones sont une puissante source d'inspiration pour la mise au point d'algorithmes nouveaux. Nous avons en particulier proposé les algorithmes suivants, qui ont donné lieu à plusieurs brevets:
   (a) La Représentation Scalaire Distribuée, qui permet la constitution de prédicteurs nettement plus puissants que les prédicteurs classiques. Cette structure, et son algorithme d'apprentissage associé, constituent également un outil de réduction de dimensionalité qui, contrairement aux méthodes antérieures, n'est pas limité au cas linéaire.
   (b) La Quantification Vectorielle avec Voisinage a été le fruit d'une étude théorique de l'algorithme de Kohonen. Nous avons montré l'apport de ce nouvel algorithme par rapport aux algorithmes classiques sur une application de compression de données.
2. Dans le domaine du traitement d'images, les réseaux neuronaux permettent de développer rapidement une solution à des problèmes très variés. Pour les différents problèmes traités en reconnaissance de formes, on retrouve globalement l'approche suivante:

(a) Réalisation d'un prétraitement réversible, ou quasi-réversible, au sens du problème à traiter. Ceci permet de réduire la complexité du problème, sans risque de perte d'information. La mise en œuvre d'un réseau de neurones autorise un prétraitement simple, ce qui évite toute perte de temps excessive dans la sélection des caractéristiques à extraire.

(b) Prise de décision par un réseau de neurones. A ce niveau, les méthodes proposées pour le réglage automatique des paramètres permettent une mise en œuvre aisée. Les réseaux de neurones conduisent généralement à de bons résultats, malgré la simplicité du prétraitement.

3. Dans le domaine du traitement du signal, nous avons montré, sur 2 applications, comment les réseaux de neurones, ou les algorithmes dérivés, autorisent la conception de solutions originales à des problèmes classiques:

   - En compression de données, nous avons proposé une structure originale, à 2 étages, qui exploite les propriétés topologiques de l'algorithme de Quantification Vectorielle avec Voisinage, et nous avons montré expérimentalement un gain de 15 à 25% par rapport à une structure plus classique.
   - En séparation de mélanges de signaux, nous avons proposé un nouvel algorithme capable de traiter des mélanges non-linéaires. Nous avons montré la robustesse de la méthode par rapport au bruitage des observations.

Du point de vue de l'industriel impliqué dans la recherche, la conclusion générale que l'on peut en tirer, est que les réseaux de neurones constituent un domaine à ne pas négliger, non seulement pour leurs capacités à résoudre rapidement (et donc à moindre coût) des problèmes très variés, mais aussi (et surtout) pour leurs potentialités en tant que source d'inspiration pour le théoricien.

Les travaux relatés dans le présent mémoire ne doivent pas pour autant être considérés comme achevés. Dans le domaine théorique, divers problèmes restent à étudier, notamment en séparation de sources (voir conclusion du chapitre). Dans le domaine applicatif, l'intégration des solutions proposées à des systèmes plus larges est actuellement en cours, notamment en classification de textures (intégration à une chaîne opérationnelle de traitement d'images), et en reconnaissance d'objets dans la cadre de la robotique.

BREVETS

References

[1] Gilles BUREL, Jean-Yves CATROS
"Procédé et dispositif de segmentation automatique d'images par analyse de texture"
Brevet n^o 89-12894 déposé le 3 octobre 1989

[2] Gilles BUREL, Isabelle POTTIER
"Procédé de compression d'images par auto-organisation d'un réseau neuronal"
Brevet n^o 89-16111 déposé le 6 décembre 1989

[3] Gilles BUREL, Isabelle POTTIER
"Procédé de classification automatique d'objets à 2 dimensions"
Brevet n^o 90-11989 déposé le 28 septembre 1990

[4] Gilles BUREL
"Procédé d'apprentissage automatique et réseau connexionniste multi-couches pour la mise en œuvre de ce procédé"
Brevet n^o 90-13445 déposé le 30 octobre 1990

[5] Gilles BUREL
"Procédé de reconnaissance automatique d'objets tridimensionnels par analyse d'image"
Brevet n^o 90-13943 déposé le 9 novembre 1990

[6] Gilles BUREL
"Procédé et dispositif automatique de séparation de sources"
Brevet n^o 91-08607 déposé le 9 juillet 1991

[7] Gilles BUREL
"Procédé de connexion interne pour réseaux de neurones"
Brevet n^o 91-11612 déposé le 20 septembre 1991

[8] Gilles BUREL
"Procédé de compression d'images"
Brevet n^o 91-11609 déposé le 20 septembre 1991

[9] Gilles BUREL, Isabelle POTTIER, Jean-Paul ACCARIE
"Procédé de reconnaissance automatique de chiffres manuscrits
dans une images et dispositif destiné à sa mise en œuvre"
Brevet n^o 91-12227 déposé le 4 octobre 1991

ARTICLES

References

[1] Gilles BUREL, Dominique CAREL, Jean-Yves CATROS
"A connectionist system for recognition of 2D workpieces"
Revue Technique Thomson-CSF, *vol* 22, n^o 4, décembre 1990

[2] Gilles BUREL, Jean-Yves CATROS
"Réseaux de neurones en traitement d'images"
Bulletin d'information des Laboratoires Centraux de Thomson-CSF, n^o 4, décembre 1990

[3] Gilles BUREL, Isabelle POTTIER
"Vector Quantization of Images using Kohonen algorithm -Theory and Implementation- "
Revue Technique Thomson-CSF, vol 23, n^o1, mars 1991

[4] Florence JACQUET, Gilles BUREL, Jean DESMOUCEAUX, Nicolas DERYCKE, Hervé NOEL
"Application des réseaux de neurones à la Veille Panoramique Infra-Rouge"
Revue Technique Thomson-CSF, vol 23, n^o1, mars 1991

[5] Gilles BUREL
"Reconnaissance d'objets 3D par réseau d'automates"
Congrès AFCET-RFIA, Lyon-Villeurbanne, 25-29 novembre 1991

[6] Gilles BUREL, Jean-Yves CATROS, Hugues HENOCQ
"Caractérisation et classification de textures sur images naturelles"
à paraître dans Traitement du Signal, vol 9, n^o1, mars 1992

[7] Gilles BUREL
"Blind separation of sources: a non-linear neural algorithm"
à paraître dans Neural Networks

SOMMAIRE

Résumé 5

Remerciements 7

Introduction 11

Chapitre 1: Données Neurobiologiques 17

1 Le neurone biologique **18**

1.1 La cellule nerveuse 18

1.2 L'influx nerveux 19

1.2.1 Le potentiel de repos 19

1.2.2 Le potentiel d'action 20

1.2.3 La propagation de l'influx nerveux 21

1.3 La transmission synaptique 23

1.4 Modélisation du neurone 24

2 Le système visuel humain **28**

2.1 Généralités 28

2.2 La rétine 29

2.3 Un relais intra-cérébral : le corps genouillé latéral 33

2.4 Le cortex visuel primaire 34

2.5 Le cortex associatif 37

2.6 Les performances du système visuel 38

2.6.1 Résultats expérimentaux 38

2.6.2 Conséquences 39

3 L'apprentissage **40**

3.1 Généralités 40

3.2 Mise en place des fibres nerveuses 41

3.3 Etablissement des connexions 42

3.4 L'ajustement de l'efficacité des connexions 44

4 Conclusion **45**

Chapitre 2: Les modèles neuronaux 49

1 L'algorithme de rétropropagation **50**

1.1 Le modèle et l'apprentissage 50

1.2 Le choix de la fonction de transition 53

2 Relations avec la théorie de la classification **56**

2.1 Généralités 56

2.2 Classifieurs Bayésiens 58

2.3 Classifieurs non probabilistes 60

2.3.1 Définition 60

2.3.2 Les classifieurs par hyperplans 60

2.3.3 Les classifieurs par hypersphères 61

2.3.4 Les classifieurs par nœuds 62

2.4 Le perceptron multicouches comme classifieur 63

3 Le modèle de Kohonen **66**

4 La quantification vectorielle **69**

4.1 Définition 69

4.2 Lien avec l'algorithme de Kohonen 72

Chapitre 3: Compléments algorithmiques 79

1 Introduction **80**

2 Réglage automatique des paramètres **82**

2.1 Initialisation des poids synaptiques 82

2.2 Facteur d'oubli du filtrage passe-bas 86

2.3 Vitesse d'apprentissage 89

3 Contrôle des facteurs déstabilisants **91**

3.1 Contrôle de la saturation 91

3.2 Limitation de la perturbation sur les potentiels 92

4 Fonction d'erreur améliorée **93**

Chapitre 4: Nouveaux algorithmes et résultats théoriques 99

1 Une nouvelle approche pour les réseaux de neurones: la Représentation Scalaire Distribuée **100**

1.1 Introduction . . . 100

1.2 La Représentation Scalaire Distribuée . . . 100

1.3 Echantillonnage de la RSD . . . 102

1.4 Limitation de la RSD à un intervalle . . . 103

1.5 Algorithme d'apprentissage . . . 104

1.6 Application à la réduction de dimensionnalité . . . 105

1.7 Application à la prédiction . . . 108

2 Nouveaux résultats relatifs à l'algorithme de Kohonen **109**

2.1 Introduction . . . 109

2.2 Rappel de l'algorithme de Kohonen . . . 110

2.3 Etat de quasi-équilibre . . . 111

2.4 Quantification Vectorielle avec notion de voisinage . . . 112

2.4.1 Idée générale . . . 112

2.4.2 Algorithme proposé (VQN) . . . 112

2.5 Quelques résultats théoriques . . . 113

2.6 Deux algorithmes de minimisation de e_{QMV} . . . 116

2.7 Forme approchée de e_{QMV} . . . 119

2.8 Quelques résultats expérimentaux en dimension 2 . . . 119

Chapitre 5: Discrimination de textures 127

1 Méthodes classiques **128**

2 Méthode proposée **129**

2.1 Idées générales 129

2.2 Vérification de la pertinence des paramètres 130

2.3 Prétraitement 133

2.4 Le réseau de neurones 135

2.5 Calcul de la confiance et post-traitement 137

3 Résultats statistiques **138**

3.1 Conditions expérimentales 138

3.2 Résultats statistiques et comparaisons 139

4 Traitement d'images complètes **141**

5 Conclusion **142**

Chapitre 6: Reconnaissance d'objets 2D 157

1 Travaux antérieurs **158**

1.1 Généralités 158

1.2 La méthode "HYPER" 159

1.3 Méthodes par transformée de Hough 161

2 Description de la méthode **163**

2.1 Généralités 163

2.2 L'approximation polygonale 165

2.3 Les fonctions de reconnaissance 167

2.4 Le réseau de neurones 169

3 Un système d'apprentissage symbolique: AGRRAPE 169

3.1 Introduction 170

3.2 La représentation des règles 171

3.3 L'apprentissage 172

3.4 La phase de reconnaissance 173

4 Classifieur de Bayes 173

5 Résultats expérimentaux 174

5.1 Conditions expérimentales 174

5.2 Comparaisons 174

5.3 Autres résultats 176

6 Conclusion 177

Chapitre 7: Reconnaissance d'objets 3D 187

1 Introduction 188

2 La base d'images et le traitement bas niveau 189

2.1 La base d'images 189

2.2 Extraction de la silhouette 190

3 Les descripteurs de Fourier **191**

3.1 Introduction 191

3.2 Normalisation 191

4 Le réseau de neurones **193**

4.1 Structure du réseau 193

4.2 Apprentissage et Classification 194

5 Résultats expérimentaux **195**

5.1 Représentation des entrées 195

5.2 Performances obtenues 196

6 Conclusion **198**

Chapitre 8: Reconnaissance de chiffres manuscrits 211

1 Introduction **212**

2 Travaux antérieurs **213**

3 Méthode proposée **214**

4 Les prétraitements **215**

4.1 Les caractéristiques métriques 215

4.2 Les caractéristiques statistiques 217

4.3 Les caractéristiques morphologiques 218

5 Le réseau de neurones **220**

5.1 Structure 220

5.2 L'algorithme d'apprentissage 224

6 Résultats expérimentaux **225**

6.1 Conditions expérimentales 225

6.2 Résultats en multi-scripteurs inconnus 225

6.2.1 Comparaison des classifieurs 225

6.2.2 Evaluation des caractéristiques 226

6.2.3 Visualisation des résultats 229

6.2.4 Résultats en fonction du seuil de rejet sur la confiance 230

6.3 Résultats en multi-scripteurs connus 230

6.4 Situation par rapport aux travaux antérieurs 231

7 Conclusion **232**

Chapitre 9: Compression d'images 237

1 Généralités **238**

1.1 Entropie d'une image 238

1.2 Techniques existantes 239

2 Expérimentations de Quantification Vectorielle **241**

2.1 Principe de la compression d'image par Quantification Vectorielle 241

2.2 La base d'images utilisée pour les expérimentations 242

2.3 Comparaison des algorithmes de Quantification Vectorielle 242

2.3.1 Objectifs 242

2.3.2 Choix de la vitesse d'apprentissage pour l'algorithme de Kohonen 244

2.3.3 Comparaison des algorithmes VQN, VQNf, LBG et k-means 244

2.4 Apprentissage adapté à l'image 246

3 Exploitation de la topologie **247**

3.1 Idée générale 247

3.2 Méthode proposée 249

3.3 Le codeur et le décodeur 249

3.3.1 Le codeur 251

3.3.2 Le décodeur 253

3.4 Résultats expérimentaux (VQN + Prédiction) 253

3.4.1 Comparaison de différents prédicteurs 253

3.5 Taux de compression obtenus 256

3.6 Variante (VQ+VQ) 257

4 Conclusion **258**

Chapitre 10: Séparation de sources 267

1 Introduction **268**

2 Travaux précédents **270**

3 Méthode proposée **273**

3.1 Idée générale 273

3.2 Elaboration d'une mesure de dépendance 275

3.2.1 Notations 275

3.2.2 Définition d'une mesure de dépendance 275

3.2.3 Expression de la mesure de dépendance en fonction des moments 276

3.2.4 Remarques 278

3.3 Définition d'une fonction de coût 279

3.4 Apprentissage en mode global 281

3.5 Apprentissage en mode continu 282

4 Résultats expérimentaux **283**

4.1 Conditions expérimentales 283

4.2 Calcul automatique de la vitesse d'apprentissage 284

4.3 Les mélanges 285

4.4 Séparation du mélange linéaire 286

4.5 Séparation d'un mélange non-linéaire en carré à 8 inconnues 287

4.6 Séparation de mélange non-linéaire bruité 290

5 Conclusion **292**

Conclusion 303

Brevets 305

Articles 306

Sommaire 309